NUCLEAR DATA, INC.
Post Office Box 451
Palatine, Illinois 60067

June, 1971

ND812 UTILITIES MANUAL

TABLE OF CONTENTS

1. INTRODUCTION

A. PROGRAM SUMMARY

The Binary Loader loads binary formatted program record into the ND812 Central Processor via the high or low speed paper tape readers or the Magnetic Tape Cassette Unit.

B. PROGRAM AREA

$76ØØ_8$ through 7753_8.

C. STARTING ADDRESS

$77ØØ_8$.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor equipped with an ASR-33 Teletype. Optional peripherals include a high speed reader and a Magnetic Tape Cassette Unit.

E. DEFINITIONS

1. STATUS WORD

This is a 12-bit binary word interpreted by the Binary Loader to determine the input device and, if necessary, the tagword of a magnetic tape cassette record. This word is either entered manually by setting it into the Switch Register before the program is started or it is loaded into the J-register when the Binary Loader is entered under software control (Software Entry).

2. PROGRAM RECORD

The program record is the collection of binary words, etc. which, when interpreted by the appropriate loader, results in the processor being loaded with machine instructions and data executable as a program. The Object Coding produced by an assembler is the commonest example of a program record. The Binary Loader discussed here is designed to interpret object coding from the BASC-12 Assembly Language Processor, program records produced by the Multiple Field Binary Writer or any other records in "Binary Format".

3. TAGWORD

This is an 8-bit character appearing after the file mark on a magnetic tape cassette program record which identifies the record to the loader. The practice allows more than one record to be written on a single cassette and random loading of any one of them.

4. FILE MARK

This is a special character written between records to separate program records on a cassette.

5. BLOCK

The block is a series of consecutive 12-bit memory addresses preceded by an origin. A program record consists of one or more blocks.

6. ORIGIN

The origin is a 12-bit word which is interpreted by the Binary Loader as an address where the first 12-bit word of a block is to be loaded. Subsequent words of the block are loaded into consecutive memory locations until another origin is detected or the end of the program record is reached. Thus, each block is preceded by one "origin".

2. PROGRAM DESCRIPTION

Because the Binary Loader is itself a program, there must be some means of loading it into memory. This is accomplished with a Bootstrap, which is a very short program loaded from the Switch Register. When executed, the Bootstrap loads enough of the Binary Loader to allow the Binary Loader to complete loading itself. The Bootstrap is destroyed in the process so that if it is necessary to reload the Binary Loader, the Bootstrap must first be reloaded from the Switch Register.

The Binary Loader begins loading a program from one of the paper tape readers by reading leader. Actual loading of the processor begins when the Binary Loader detects the first character different from 0200_8 (eight level punch only). For this reason, it is essential that a program tape be placed in the reader with the leader at the read station. Should the program tape be placed in the reader with blank tape at the read station, the Binary Loader begins loading zeros into memory beginning with location ØØØØ$_8$. Blank tape is not a 200 level and the loader assumes that it must begin loading another program. The actual leader of the program record is interpreted as **trailer.**

The loading process consists of assembling consecutive pairs of frames using levels one through six as high and low order halves of 12-bit words. The assembled 12-bit words are stored in the processor in consecutive memory locations as determined by the presence and interpretation of the origins on the paper tape. Field change characters cause the storage to take place in memory fields specified by the last 2 bits of the characters. When a program record is created, the last two characters are written in such a way as to make the sum of all the 12-bit words (including the last) on the tape equal to zero. The loader keeps a running sum or checksum of the 12-bit words in the record. When the loader detects the trailer, it tests the checksum and if it is zero, the loading process is assumed to be correct and the loader stops with the J-register equal to zero. If the checksum is non-zero, the loading process is assumed to be in error. In any case, the checksum is left in the J-register when the program stops.

The following describes the various parts of Figure 1:

1) Leader/Trailer - Leader and trailer are punched identically and indicate, respectively, the beginning and end of a binary format program record on paper tape. Program records written on magnetic tape use a file mark and tagword instead of leader.

2) Field Change Character - A special character created by the FIELD directive in the BASC-12 Assembly language Processor used to indicate the field in which the program is to be stored as it is loaded. If this character does not appear on the tape, the loader will load into the same field in which it itself is located. Levels eight and three are punched to indicate to the loader a field change and levels one and two are used to indicate the new field (Ø, 1, 2 or 3). Levels four through seven must not be punched.

3) Origin - A 12-bit word punched in two frames of six bits each interpreted by the Binary Loader as an address at which to begin storing programs. Origins are generated on a program source tape with the use of the special character asterisk (*). An origin is distinguished from program words on the binary tape by the presence of the seven level punch. Levels one through six are the higher 6-bits of the address. The low order 6-bits may be found in levels one through six of the next tape frame.

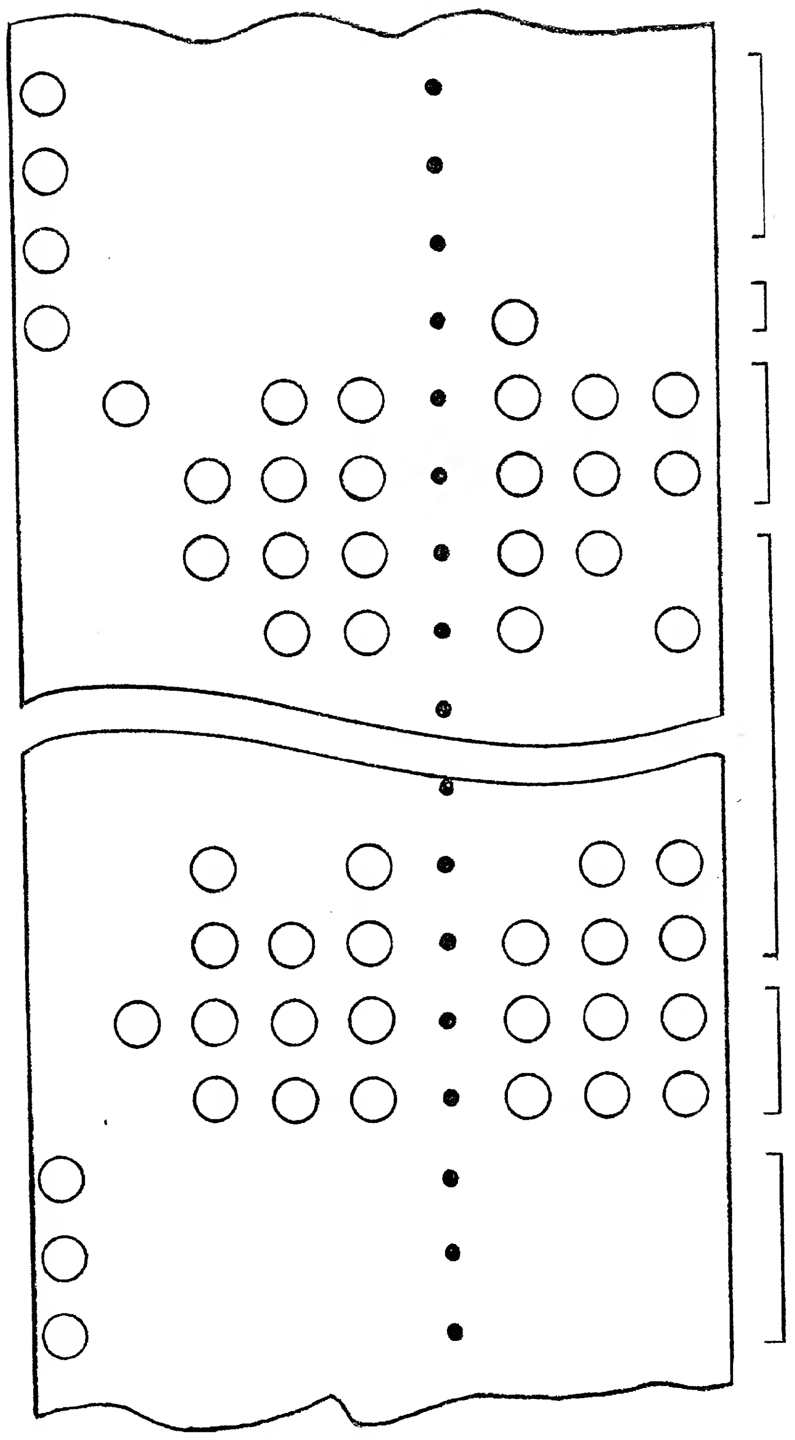

LEADER
(8th level only)
FIELD CHANGE
(8th and 3rd levels with 1st and 2nd indicating field)
CRIGIN
PROGRAM
CHECKSUM
TRAILER

FIGURE 1

4) Program - The program consists of 12-bit data words which are to be stored in the processor for execution as the user's program. The 12-bit data words appear on the tape as two consecutive frames of 6-bits each. The seven and eight levels must not be punched.

5) Checksum - This is a running sum of all the 12-bit words on the tape, including the program words and origin, excluding field change characters and the checksum itself. Thus, the sum of all the 12-bit words on the tape including the checksum should be zero. If the result is not zero, the program was not correctly loaded and must be reloaded by repeating the operational steps of the Binary Loader. The checksum is always written in two halves as the last two frames in the record and is further indicated by a 7-level punch on the first half.

There are three alternative methods of using the Binary Loader. In all cases the format of the record being loaded remains the same. The differences lie in the manner of entering the Binary Loader and exiting once the loading process has been completed. In all three of the cases to be described, location 7751_8 contains the exit address. That is, instead of stopping when the loading process is complete, the Binary Loader performs a jump to the address contained in location 7751_8, provided the contents of location 7751_8 is not zero. This feature will be described as the Auto-start feature of the Binary Loader.

Manual Load with Auto-start is the simplest use of the Auto-start feature of the Binary Loader. The program record being loaded includes the necessary coding to cause location 7751_8 to be set equal to some non-zero address. When the Binary Loader completes the loading process, it will jump to this address with the checksum, (normally zero), in the J register and the exit address in the K register to determine if the loading process was correct and take appropriate action should the J register be non-zero.

Second use of the Auto-start involves performing a JPS to the Binary Loader from a program outside the Binary Loader. The JPS is performed to location 7751_8. When the loading process is completed the Binary Loader will return to the calling location plus one with the checksum in the J register, as though the Binary Loader were a sub-routine. The program being loaded must not alter location 7751_8. When entering the Binary Loader in this fashion, it is necessary that the status word, which is normally loaded from the switch register, be loaded in the J register by the calling program before the JPS to the Binary Loader is executed.

The third method of using the Auto-start feature is very similar to the second method but rather than a JPS, the calling program performs a JMP to location 7752_8. Again, the status word must have been loaded into the J register by the calling program. It is necessary that the calling program either set location 7751_8 to zero, thereby causing the Binary Loader to stop at the end of the loading process (provided the program being loaded does not alter the location 7751_8) or the calling program should set an appropriate exit address into location 7751_8.

3. OPERATOR OR USER CONTROL

The status word is the only control the user has over the loader. The bits of the status word are interpreted by the loader as follows: BITS Ø and 1 determine the input device, BITS 2, 3 and 4 determine the input cassette drive if the cassette was selected, and BITS 5 through 11 indicate the tagword when loading from magnetic tape cassette. Each one of these three functions is described in detail below.

1) Input Device Selection - BITS Ø and 1 determine the device from which the record is to be read. If BIT 1 is "Ø", input is from cassette and BIT Ø is ignored. If BIT 1 is "1", input is from one of the paper tape readers as controlled by BIT "Ø". When BIT Ø is "Ø", the input is from the high speed reader and if BIT Ø is "1", the input is from low speed or teletype reader.

2) Cassette Drive Selection - BITS 2, 3 and 4 permit the user to select one of the three cassette drives for input. No two of of these bits should be on together as the loader will try to read from two or more drives simultaneously. BIT 2 set to "1" selects drive three, BIT 3 selects drive two, and BIT 4 selects drive one.

3) Tagword Selection - Any tagword from $ØØØØ_8$ to $Ø177_8$ may be selected by BITS 5 through 11. Capacity to select a tagword allows the user to select at random one of many records on a particular cassette without the need to hunt manually for the record in question. Starting with the beginning of the cassette, the Binary Loader will search for the correct tagword by spacing forward to file mark and reading the tagword. The Binary Loader will continue to space forward and read the tagword until the selected tagword is found.

NOTE

Entering the Binary Loader under software control demands that the J register be set to the desired status word by the software performing the call to the Binary Loader (as described above).

4. OPERATIONAL PROCEDURE

The following is the procedure by which the Bootstrap is loaded in the ND812.

1) Place the POWER ON/POWER OFF/CONTROL OFF switch in the POWER ON position.

2) Set the Switch Register to 7762_8 and depress LOAD AR.

3) It is now necessary to load fourteen instruction from the Switch Register, each of which is followed by depressing the LOAD MR key.

ADDRESS	INSTRUCTION	ADDRESS	INSTRUCTION
7762	7404	7771	7404
7763	6101	7772	6101
7764	7403	7773	7403
7765	1146	7774	1122
7766	1501	7775	5700
7767	6105	7776	6114
7770	1101	7777	7745

4) To check if the instructions were stored in the proper locations, set the Switch Register to 7762_8 and depress LOAD AR. Place the SELECT REGISTER switch in the ADDRESS powition. Depressing the NEXT WORD key causes the SELECTED REGISTER indicator lamps the contents of the address. Continue to depress the NEXT WORD key until all instructions have been checked.

5) Set the Switch Register to 7773_8 and depress LOAD AR.

6) Place a paper tape of the ND41-0005 Binary Loader into low speed reader taking care that leader appears at the read station, turn teletype to LINE and reader ON.

7) Depress START.

The paper tape is read and will stop on reaching trailer. If the J register is zero after reading the paper tape, no errors occurred. Repeat the above from step 2 if J register is non-zero.

The Binary Loader is now in memory and is used to load program records from either paper tape or cassette.

LOW SPEED PAPER TAPE

1) Set the Switch Register to $77ØØ_8$ and depress LOAD AR.

2) Place the binary formatted program tape to be read into the ASR-33 Reader with the leader at the read station taking care that leader appears at the read station.

3) Place the START/STOP switch on the ASR-33 Reader to START.

4) Depress START.

5) The Binary Loader will read the program tape and stop at trailer. If the content of the J register is zero at completion of the loading process, the program tape was loaded correctly. If the J register is non-zero, re-start this operational procedure from Step 2.

HIGH SPEED PAPER TAPE

1) Set the Switch Register to $77ØØ_8$ and depress LOAD AR.

2) Set Switch Register BIT Ø to "Ø".

3) Place the binary formatted program tape to be read into the high speed reader with the leader at the read station.

4) Depress START.

5) The Binary Loader will read the program tape and stop at trailer. If the content of the J register is zero at completion of the loading process, the program tape was loaded correctly. If the J register is non-zero, re-start this operational procedure from Step 2.

MAGNETIC TAPE CASSETTE PROCEDURE

1) Set the Switch Register to $77ØØ_8$ and depress LOAD AR.

2) Set Switch Register BITS Ø and 1 to "Ø", set BITS 2, 3 and 4 for the desired cassette drive, set BITS 5 through 11 to desired tagword.

3) Depress START.

4) The Binary Loader will space forward to file mark and read the tagword, and continue to do so until selected tagword is found. When the tagged record is reached, the Binary Loader will read the record and stop at completion. If the content of the J register is zero, the loading process was correct. If the content of the J register is non-zero, re-start from Step 2.

NOTE

If the Binary Loader proceeds to the end of tape without stopping, the user has specified a tagword which does not exist on the cassette. To recover from this condition, depress STOP and re-start from Step 1, and ascertain that the tagword specified exists on the cassette.

5. ERROR DIAGNOSTICS

The checksum is stored in the J register at the completion of either a manual or software controlled program loading procedure. A non-zero J register indicates an erroneous load. Refer to the section of the OPERATIONAL PROCEDURE appropriate to the input devices and re-load the program. If an error is encountered under software control, check the calling program and the exit address of the Binary Loader.

If a non-existent input device is specified, the Binary Loader will enter an endless loop. The processor must be stopped with the front panel STOP switch and the loader re-started.

Failure to load a tape with leader over the read station will cause the loader to stop on reaching the program's leader. The J register will be zero, but since the program was not read, it will not be loaded. The user is particularly warned against placing blank tape at the read station when loading overlays, as part of the background program is likely to be lost when the loader tries to load the blank tape.

6. COMMAND SUMMARY

STATUS WORD INPUT DEVICE	BIT Ø	BIT 1
Cassette	Ø	Ø
High Speed Paper Tape Reader	Ø	1
Low Speed Paper Tape Reader (ASR 33)	1	1

CASSETTE DRIVE	BIT 2	BIT 3	BIT 4
DRIVE 1	Ø	Ø	1
DRIVE 2	Ø	1	Ø
DRIVE 3	1	Ø	Ø

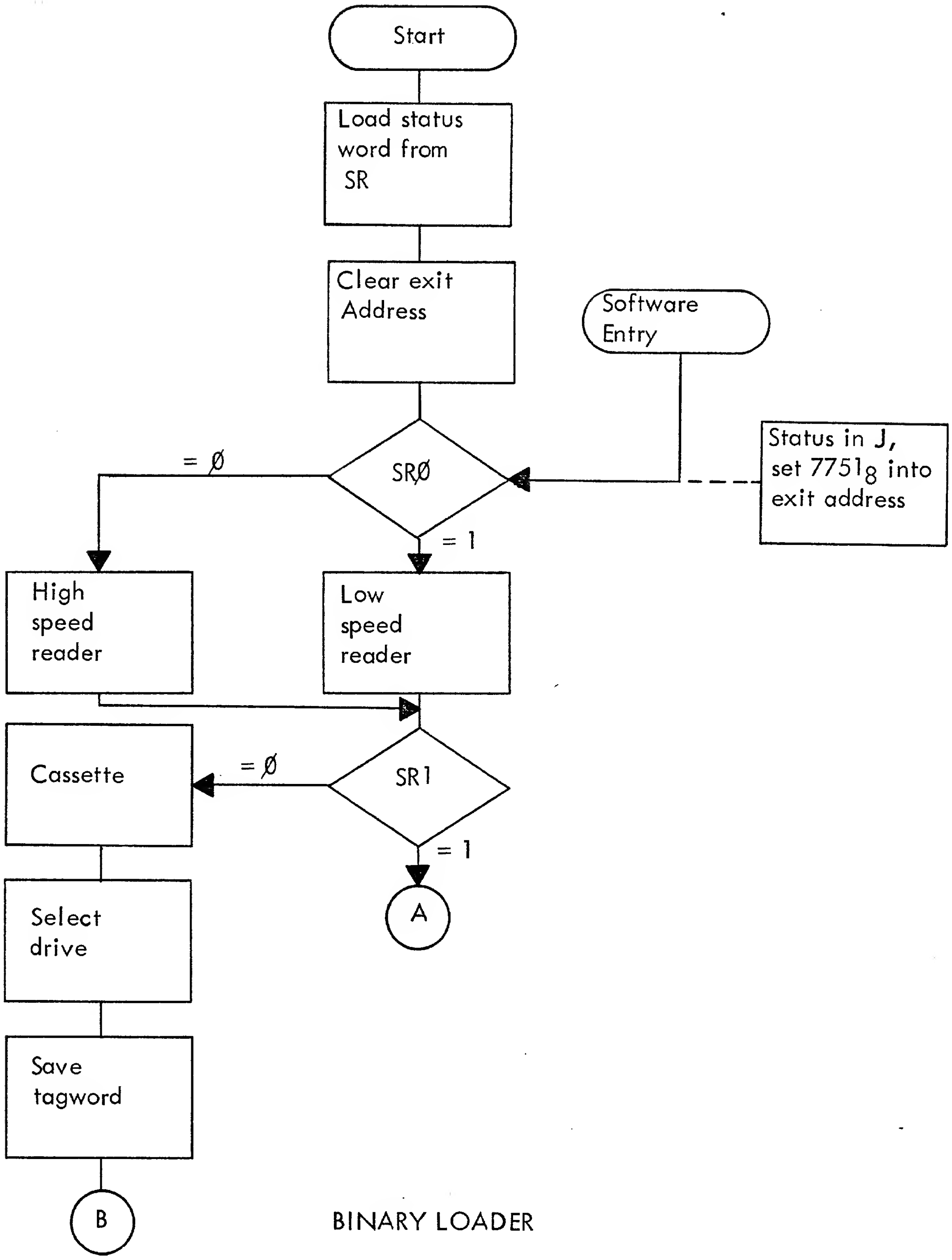
BITS 5 through 11 are used to specify the tagword.

7. FLOW CHART

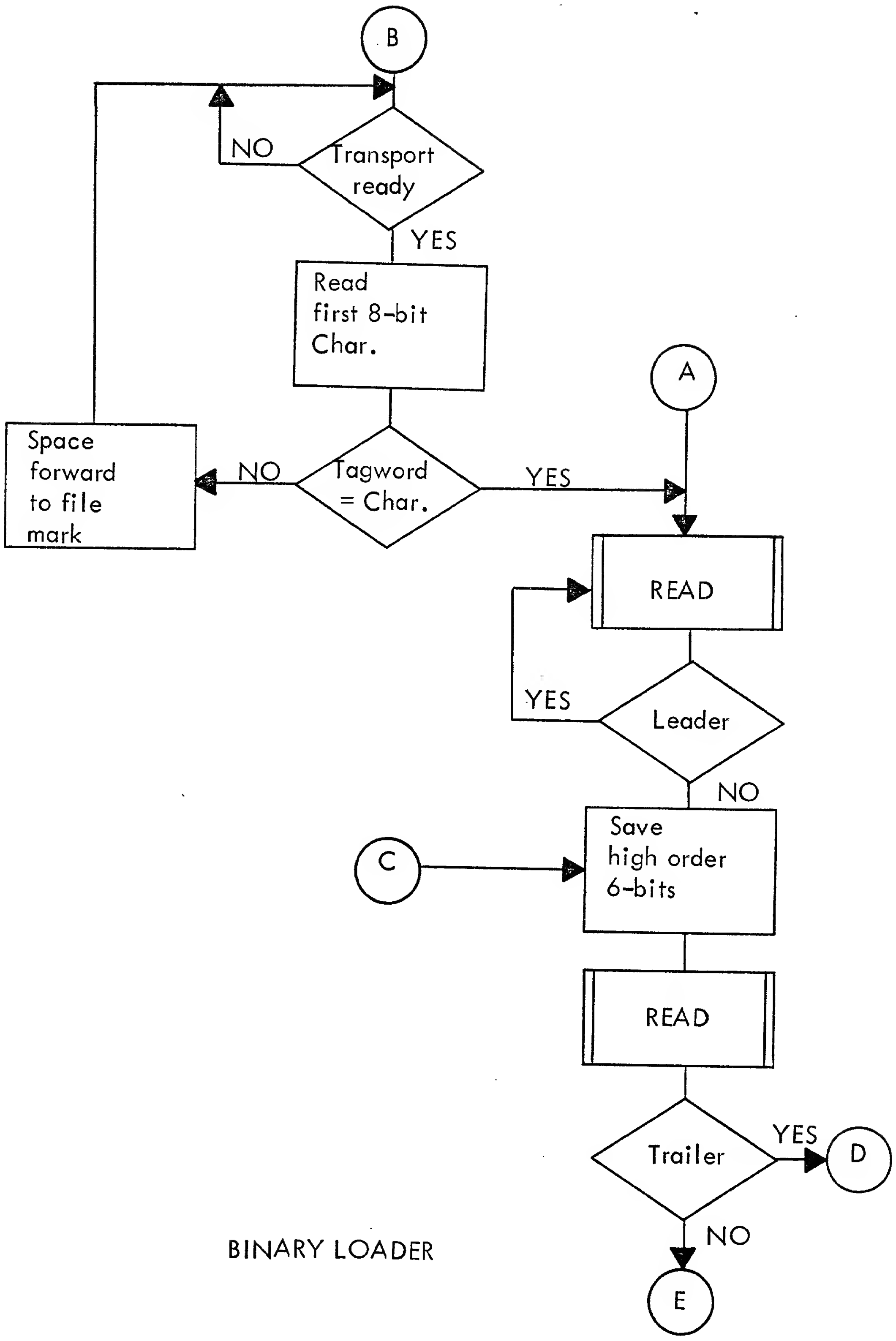
(Next Page)

8. PROGRAM LISTING

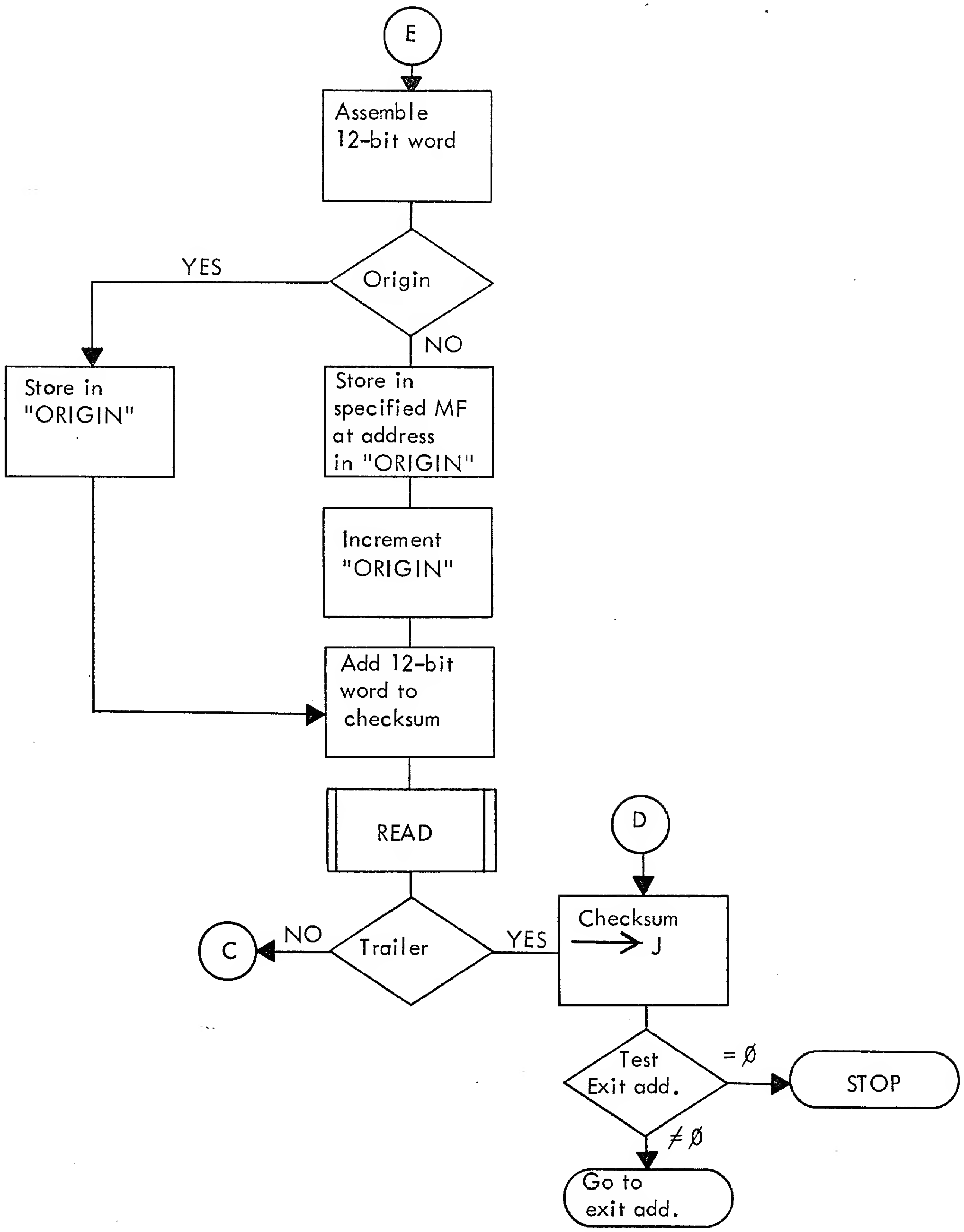
Not Applicable.



BINARY LOADER



BINARY LOADER


E
Assemble 12-bit word
Origin
YES
NO
Store in "ORIGIN"
Store in specified MF at address in "ORIGIN"
Increment "ORIGIN"
Add 12-bit word to checksum
READ
D
Trailer
NO
C
YES
Checksum → J
Test Exit add.
= Ø
STOP
≠ Ø
Go to exit add.

1. INTRODUCTION

A. PROGRAM SUMMARY

This program contains programmable subroutines that provide fast double precision addition, subtraction, multiplication, division and I/O routines.

B. PROGRAM AREA

Ø4ØØ8 through 115Ø$_8$.

C. STARTING ADDRESS

The subroutines are called by software commands from the main program.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor. An ASR-33 Teletype is needed if input or output is desired.

E. DEFINITIONS

None.

2. PROGRAM DESCRIPTION

This program contains a two-word (24 bit) Integer Accumulator (locations Ø46Ø$_8$ and Ø461$_8$) and a two-word operand holding register (Ø456$_8$ and Ø457$_8$) that make up an intricate section of the instruction interpreter. The interpreter uses a format modeled after the standard MRI's i.e., a 4-bit instruction code, an indirect bit, a direction bit and 6 address bits.

The interpreter is entered by a "JPS" to $Ø4ØØ_8$. Thereafter, the Integer Package will interpret the contents of successive memory locations following the "JPS $Ø4ØØ_8$" as instructions. A "ØØØØ" instruction causes an exit from the package and return of control to the machine instruction following the "ØØØØ". Prior to exiting with the "IEXT" (octal equivalent = ØØØØ), only those instructions listed as appropriate to the integer package operations should be used. Although use of the BASC-12 language instructions is possible while in Interpreter Mode, the results will not correspond to the BASC-12 mnemonics, it is necessary to execute and "IEXT" instruction. To revert back to integer coding, it is necessary to perform a "JPS $Ø4ØØ_8$".

Note that the Integer Package must reside in the same memory field from which it is called.

The instructions described below are to be used according to the rules for Single-word Memory Reference Instructions. There are no two-word instructions and all instructions except DINPUT and DOUTPT require an address (for the operand). Operands are expected to be in two locations; at the effective address and the effective addresses of the instructions. The effective address is found in the same manner as the effective address of a Single-word Memory Reference Instruction. Operands are stored in two consective memory locations with first location containing the 12 low-order bits and the next location containing the high 11 bits and sign. The sign convention used is Bit Ø = Ø for positive numbers, Bit Ø = 1 for negative numbers with the remaining 23 bits containing the two's complement of the number. The first word (low order bits) is tagged for reference by the Integer Instructions.

DILOD - Transfers the operand located at the effective address of the instruction to the Integer Accumulator. Data is left in memory.

DISTR - Transfers the Integer Accumulator to the effective address of the instructions. Data remains unchanged in the Integer Accumulator.

DIADD - The operand at the effective address of the instruction is added to the Integer Accumulator. Data in memory is unchanged and the overflow bit is set if the magnitude of the result is greater than 8. The sign of the result is arithmetically logical.

DISUB - The operand at the effective address of the instruction is subtracted from the Integer Accumulator. Overflow cannot occur. The sign of the result is arithmetically logical.

DIMULT - The Integer Accumulator is multiplied by the operand. The low order 23 bits of the result are left in the Integer Accumulator and the sign is restored to the highest order bit according to arithmetic logic.

The algorithm used for the multiply instruction first determines the sign of the result from the signs of the operands and stores this information. The hardware multiply instruction is then used to multiply first the low order 12 bits of the operand by the low order 12 bits of the Integer Accumulator. Next, high order 12 bits of the operand are multiplied by the low order 12 bits of the Integer Accumulator and the low order 12 bits of the results are added to the high order 12 bits of the first result. Finally, the low order 12 bits of the operand are multiplied by the high order 12 bits of the Integer Accumulator and the low 12 bits of the results are added to the high order 12 bits of the previous sum. This total is placed in the Integer Accumulator and the sign (Bit Ø) is set to conform to the value previously stored.

DIDIV - Divides the operand into the Integer Accumulator. The algorithm consists of 23 subtractions and integer accumulator left shifts. The operand is the divisor, the integer accumulator the dividend and the quotient is shifted into the Integer Accumulator from the right. The fractional part of a quotient and/or the remainder is lost. An attempt to divide by zero causes $Ø731_8$ to set = 1 and the division routine is by-passed.

DINPUT - Reads a number from the teletype, converts it to binary format (23 bits of data plus one sign bit) and leaves the result in the Integer Accumulator. Values can range from -8388608 to +8388607. Input is terminated by an 8th digit or a non-numeric character. A "RUBOUT" re-initializes the input routine and echoes a back arrow.

DIOUTPT - Performs a running Binary to BCD conversion of the Integer Accumulator contents and types the result on the teletype in an 8 column field where the first column maybe a minus sign. Leading zeros are suppressed as is the sign if the number is positive. If the number is negative, the minus sign is printed in the first column.

3. OPERATOR OR USER CONTROL

The following is a sample program illustrating use of the package.

```
4ØØØ      Ø64Ø      TWJPS
4ØØ1      Ø4ØØ      Ø4ØØ
4ØØ2      5ØØ6      DILOD   A   (1)
4ØØ3      44Ø7      DIADD   B   (2)
4ØØ4      7Ø1Ø      DIMLT   D   (3)
4ØØ5      5411      DISTR   D   (4)
4ØØ6      ØØØØ      DIEXIT
4ØØ7      ØØØØ      STOP
4Ø1Ø      ØØØØ      A,Ø
4Ø11      ØØØ2        2
4Ø12      ØØØØ      B,Ø
4Ø13      ØØØ4        4
4Ø14      ØØØØ      C,Ø
4Ø15      ØØØ6        6
4Ø16      ØØØØ      D,Ø
4Ø17      ØØØØ        Ø
```

In the preceding sample program the interpreter is entered with a two-word JPS, A is loaded into the integer accumulator, B added to it, multiplied by C and stored in D. The interpreter is then exited and the program stops. The following table contains the status of HORD and LORD (the Integer Accumulator for each instruction in the sample routine.

STEP	1	2	3	4
HORD	Ø	Ø	Ø	Ø
LORD	2	6	44	44

4. OPERATIONAL PROCEDURE

1) Load the Integer Package (ND41-0017) program tape with Binary Loader (refer to Binary Loader (ND41-0005) for description of the loading procedure.

2) Call desired routine from main program.

5. ERROR DIAGNOSTICS

A division by zero will cause the contents of location $Ø731_8$ to be incremented by one (initially set to Ø) and the division routine to be bypassed.

6. COMMAND SUMMARY

INSTRUCTION CODE	INSTRUCTION	DESCRIPTION
0000	DIEXIT	This instruction exits the interpreter mode.
3000	DINPUT	This instruction causes a JPS to the input routine. A number is read in from the keyboard and stored in the integer accumulator. This number can range from -8388608 to +8388607.

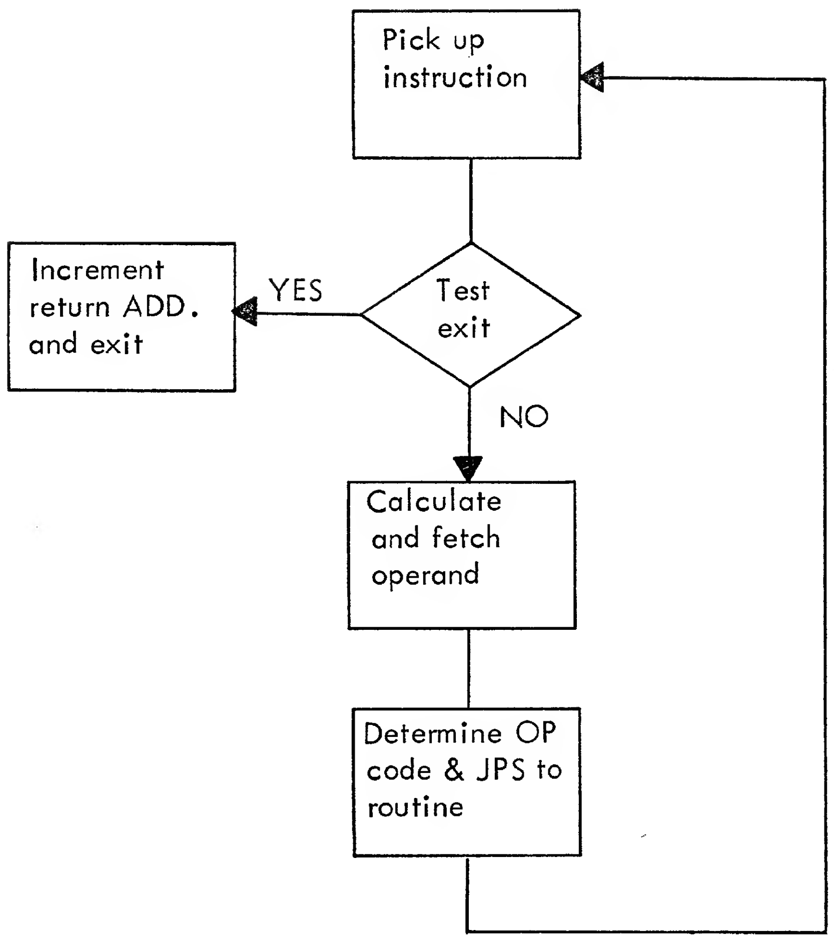
INSTRUCTION CODE	INSTRUCTION	DESCRIPTION
3400	DOUTPT	This instruction causes a JPS to the output routine and destructively outputs the number in the Integer Accumulator. The form of the output is a sign and seven digits. The number can range from -8388608 to +8388607
4000	DISUB	This instruction causes a double precision subtraction of the operand from the Integer Accumulator. The results are left in the Integer Accumulator.
4400	DIADD	This instruction causes a double precision addition of the operand to the Integer Accumulator. The results are left in the Integer Accumulator.
5000	DILOD	This instruction loads the operand into the Integer Accumulator.
5400	DISTR	This instruction stores the Integer Accumulator in the effective address. The contents of the Integer Accumulator are left unchanged.
6400	DIDIV	This instruction causes a division of the Integer Accumulator by the contents of the effective address. The quotient is left in the Integer Accumulator. Division by zero causes location $Ø731_8$ to be set $\neq 0$ and the division is bypassed.
7000	DIMLT	This instruction causes the Integer Accumulator to be multiplied by the contents of the effective address. Note that only the low order 24 bits are generated.

7. FLOW CHART

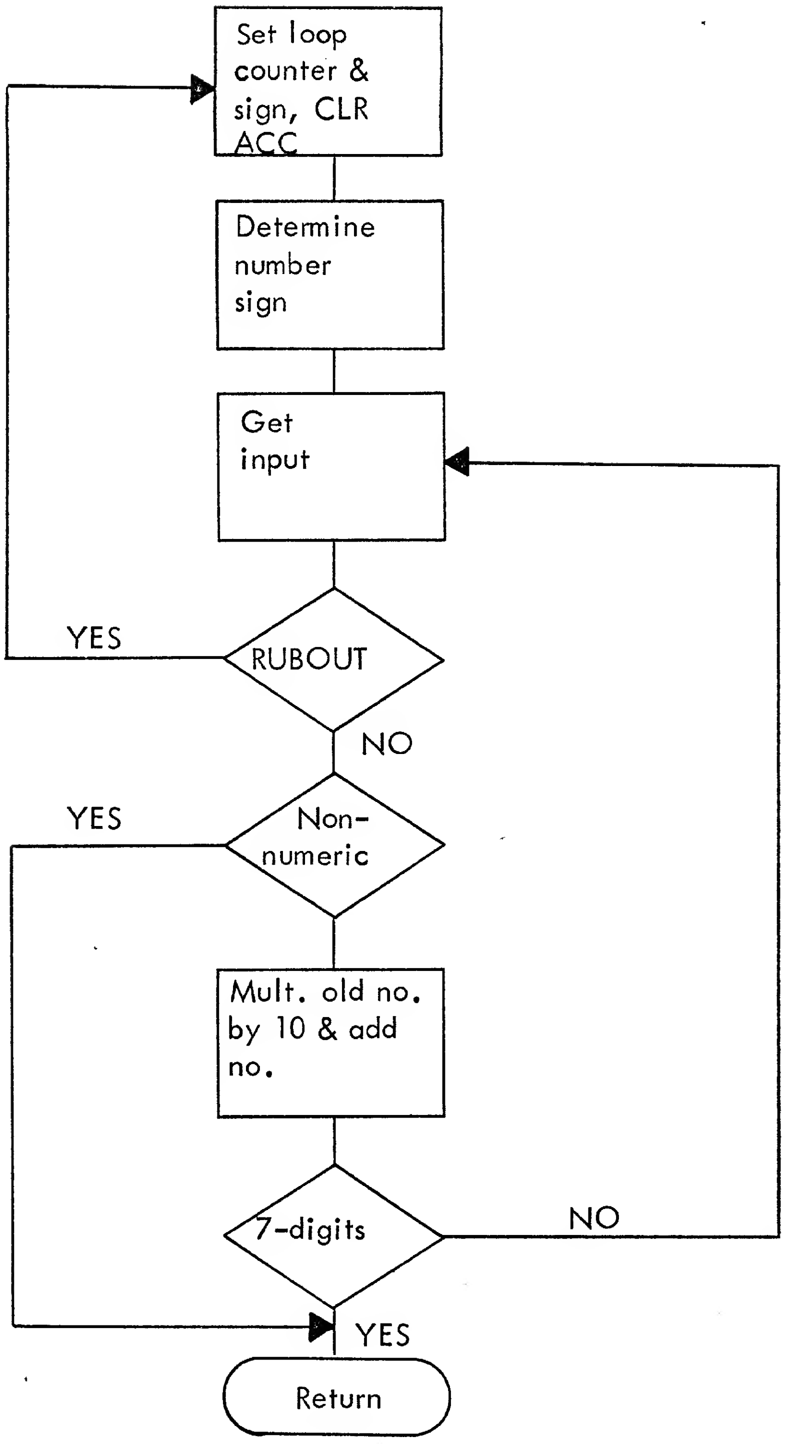
(Next Page)

8. PROGRAM LISTING

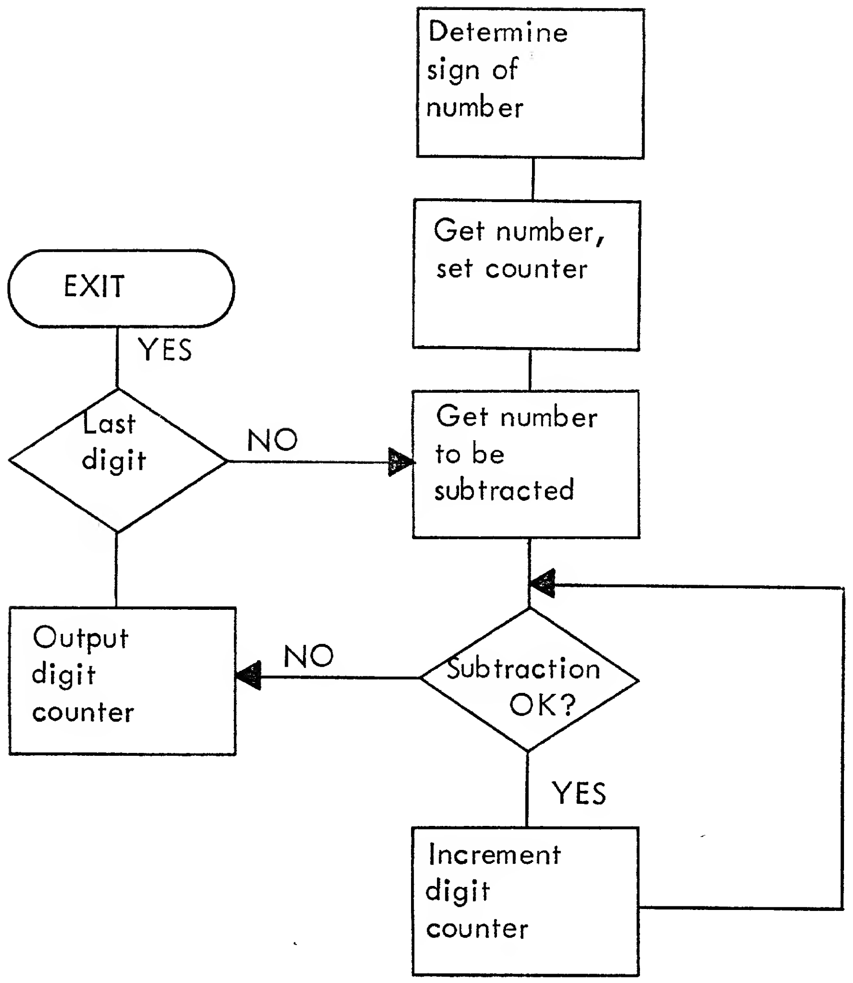
Not Applicable.


Pick up instruction
Test exit
YES
Increment return ADD. and exit
NO
Calculate and fetch operand
Determine OP code & JPS to routine

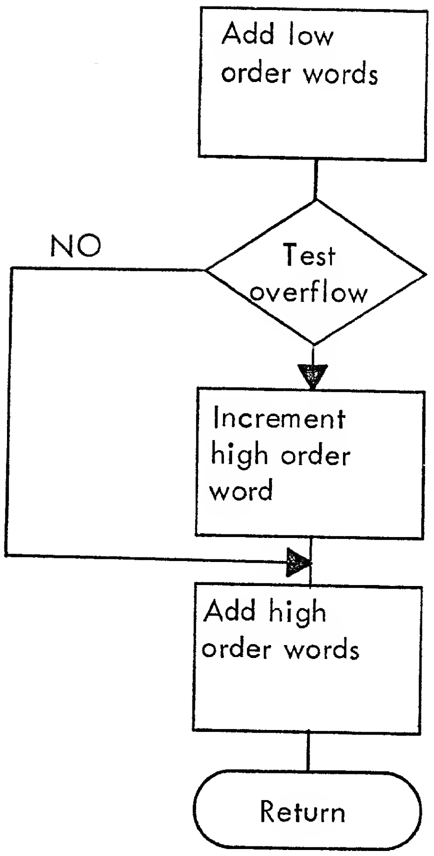
INTERPRETER



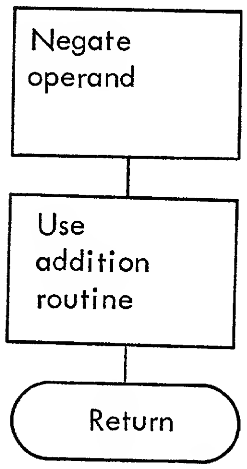
INPUT ROUTINE



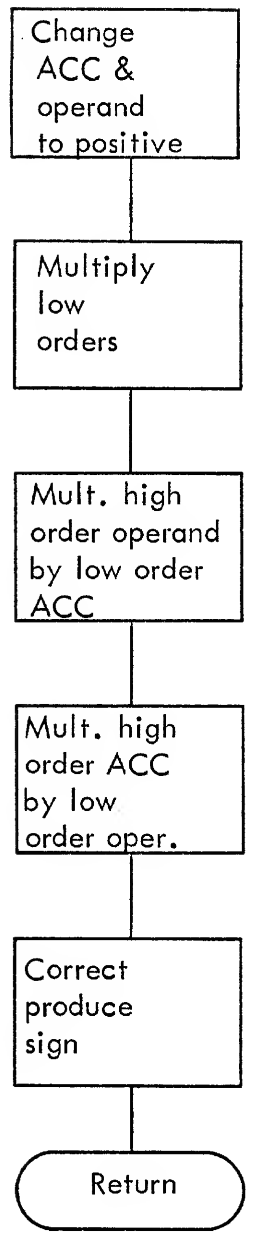
OUTPUT ROUTINE



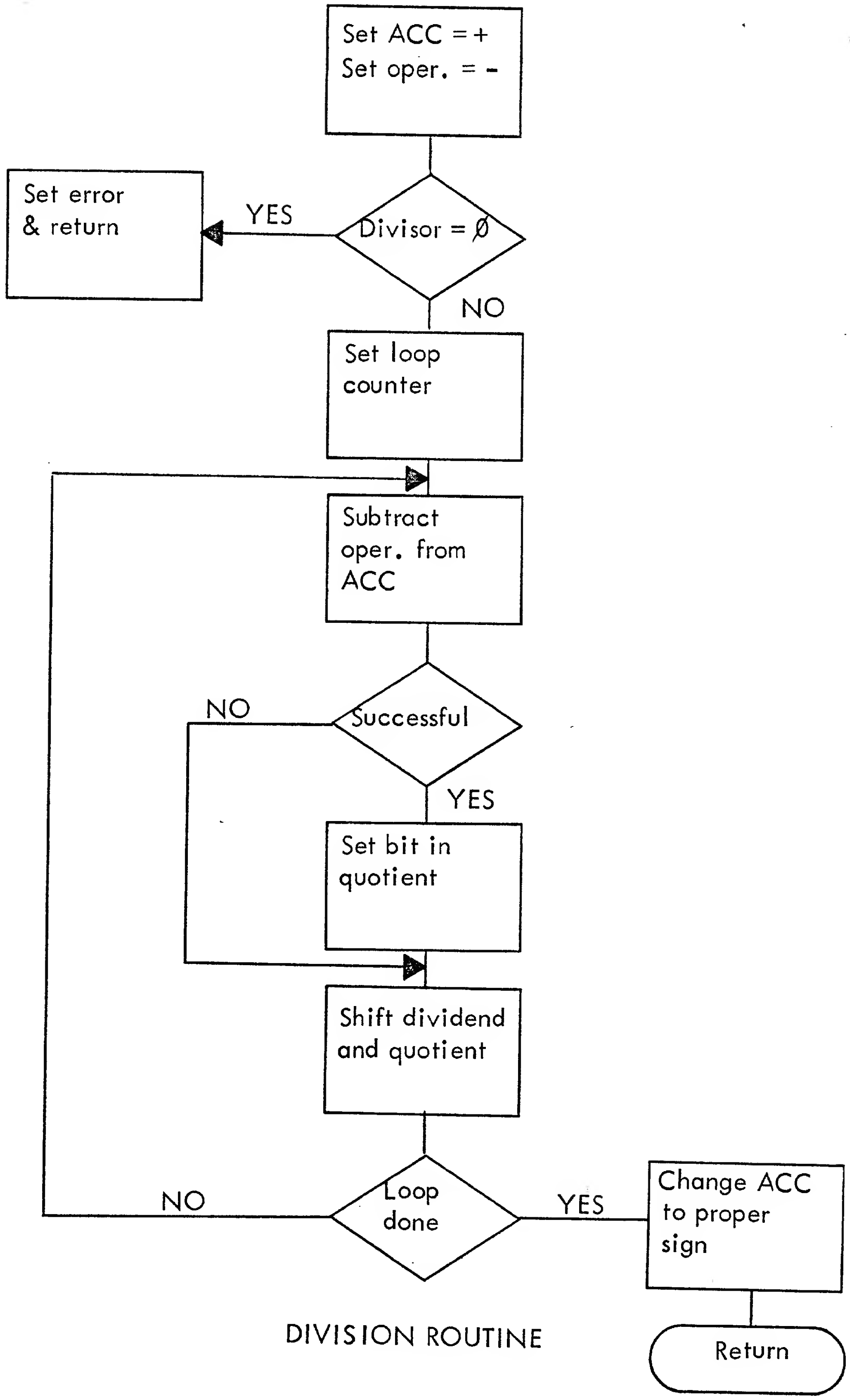
ADDITION ROUTINE



SUBTRACTION ROUTINE


Change ACC & operand to positive
Multiply low orders
Mult. high order operand by low order ACC
Mult. high order ACC by low order oper.
Correct produce sign
Return

MULTIPLICATION ROUTINE



DIVISION ROUTINE

1. INTRODUCTION

A. PROGRAM SUMMARY

The Binary Tape Copier Program duplicates and verifies binary formatted paper tapes.

B. PROGRAM AREA

The program resides in core locations $ØØØ1_8$ through $Ø322_8$ and uses locations $Ø323_8$ through 7577_8 as a buffer.

C. STARTING ADDRESS

$ØØØ1_8$ for read and punch, $ØØ22_8$ punch only, $Ø132_8$ verify only.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor equipped with an ASR-33 Teletype. Optional peripherals are high speed punch and/or reader.

2. PROGRAM DESCRIPTION

This program reads the punched binary tape and stores the data in a packed 12-bit format (two 6-bit bytes from the tape). Storage starts at location $Ø323_8$ and continues until terminated by the trailer (8th level punch). Every tape read is given an Origin which are stored in memory. Location 7577_8 will contain the first Origin, the second location 7576_8, and decrementing one location for each successive Origin. After all binary tapes are read, the program reads the Switch Register to obtain the number of duplicates and starts duplication. A checksum is kept of the output data and each duplicate contains a reader, trailer, and a separation of blank tape.

If field change characters are present on the binary tape the program stores the first field change character in location 7577_8, and decrementing one location for each successive field change character. The Origin pointer and previously stored Origins are shifted one location down for each field change character. A field change character does not update the checksum.

Verification is accomplished by reading the duplicate and keeping a running checksum. The running checksum is the added to the program checksum and is the result is non-zero the program enter on error condition and the program stops.

3. OPERATOR OR USER CONTROL

The number of duplications is determined by status of Switch Register Bits 2 through 11. These switches represent binary increments with BIT 11 weighted as one. To select high speed reader set BIT Ø to "1", or low speed reader set BIT Ø to Ø. To select high speed punch set BIT 1 to "Ø". For each verification the user must depress CONTINUE. Refer to Starting Addresses for various functions.

4. OPERATIONAL PROCEDURE

1) Load program tape with the Binary Loader (refer to Binary Loader (ND41-0005) for a detailed description of its use).

2) Place blank paper tape in punch and turn punch ON. Place tape to be duplicated in desired reader, making sure that blank tape or leader (8-level punches) appear at the read station.

3) Set Switch Register to $ØØØ1_8$.

4) Depress LOAD ADDRESS.

5) Set Switch Register BITS 2 through 11 to number (octal) of duplicates desired.

6) Set Switch Register BIT Ø to "1" for high speed reader or to "Ø" for low speed reader.

7) Set Switch Register BIT 1 to "1" for high speed punch or to "Ø" for low speed punch.

8) Depress START.

9) The program will now read the users tape, punch the number of duplicates indicated by the Switch Register, and stop.

10) To verify duplicate tapes, place the duplicate in the reader, making sure that blank tape appears at the read station. Depress CONTINUE. The program will read and verify the entire tape. If the tape is acceptable, the program will stop when the end of trailer is at the read station. Repeat for each tape to be verified.

5. ERROR DIAGNOSTICS

When an error occurs, a single letter is typed on the teletype and the program STOPS. The J register always contains the ASCII code of the letter typed (shown in parenthesis below) should any confusion arise. Should an error occur, it will be necessary to restart the program at the starting address appropriate to the mode in which the program was operating when the error occurred.

The following is a table of possible errors, their cause and correction.

ERROR	CAUSE	CORRECTION
S(323)	Start error. Did not start on blank tape or leader.	Re-start program from Step 1 of the Operational Procedure. Be sure to place blank tape or leader (8-level) over read station.
R(322)	Reader error.	Re-start program from Step 1 of the Operational Procedure.
O(317)	Buffer overflow.	Program or programs for duplication too large for buffer. Cannot be duplicated with this program.
V(326)	Verification error.	Possible punch error. Re-verify tape, if error occurs again, dispose of tape.

6. COMMAND SUMMARY

The program is controlled via the Switch Register which must be set up before the program **is started.**

BITS 2 through 11 determines number of copies produced in the duplication mode.

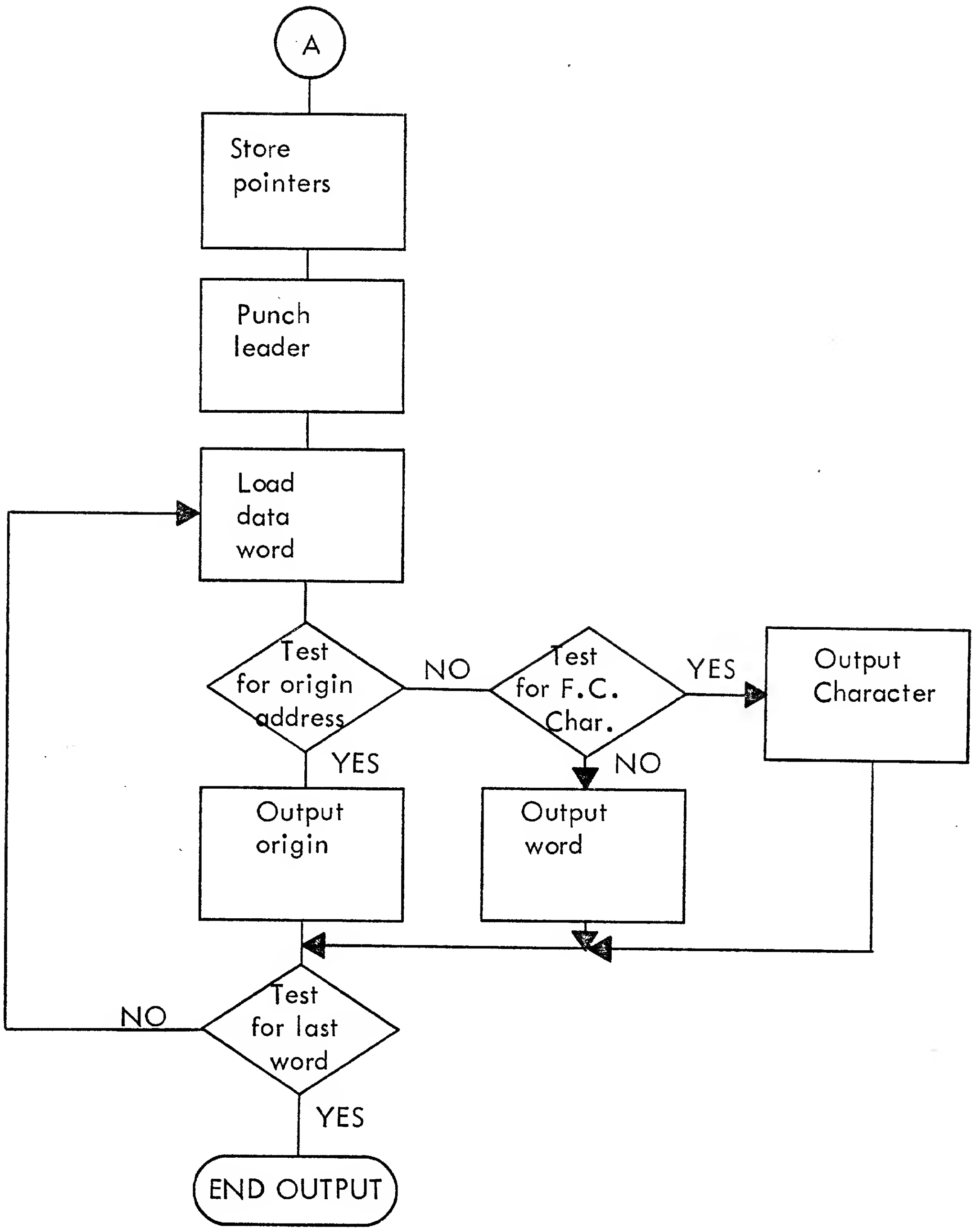
INPUT	BIT Ø	OUTPUT	BIT 1
Low speed reader (tele-type)	Ø	Low speed punch (tele-type)	Ø
High speed reader	1	High speed punch	1

7. FLOW CHART

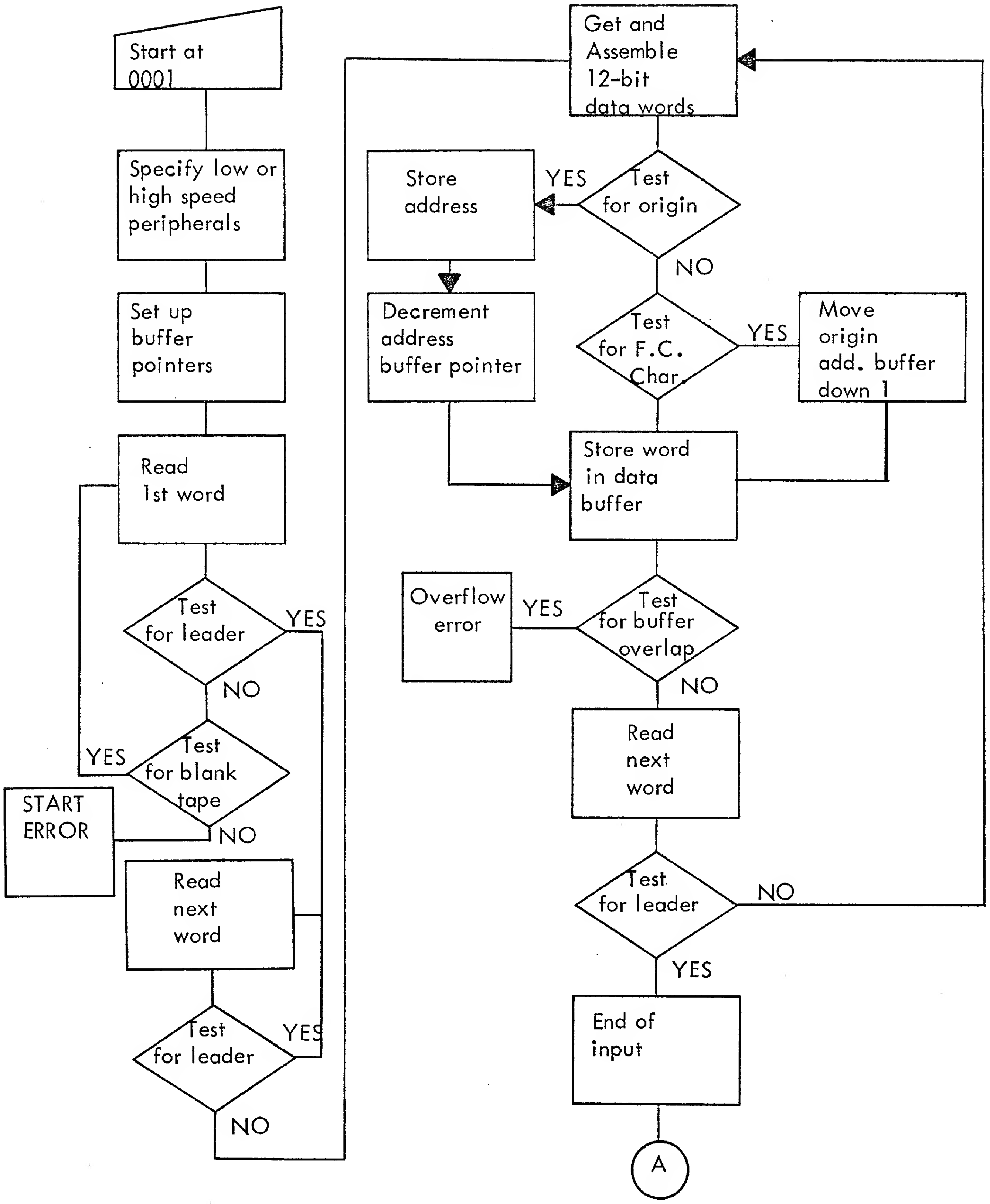
(Next Page)

8. PROGRAM LISTING

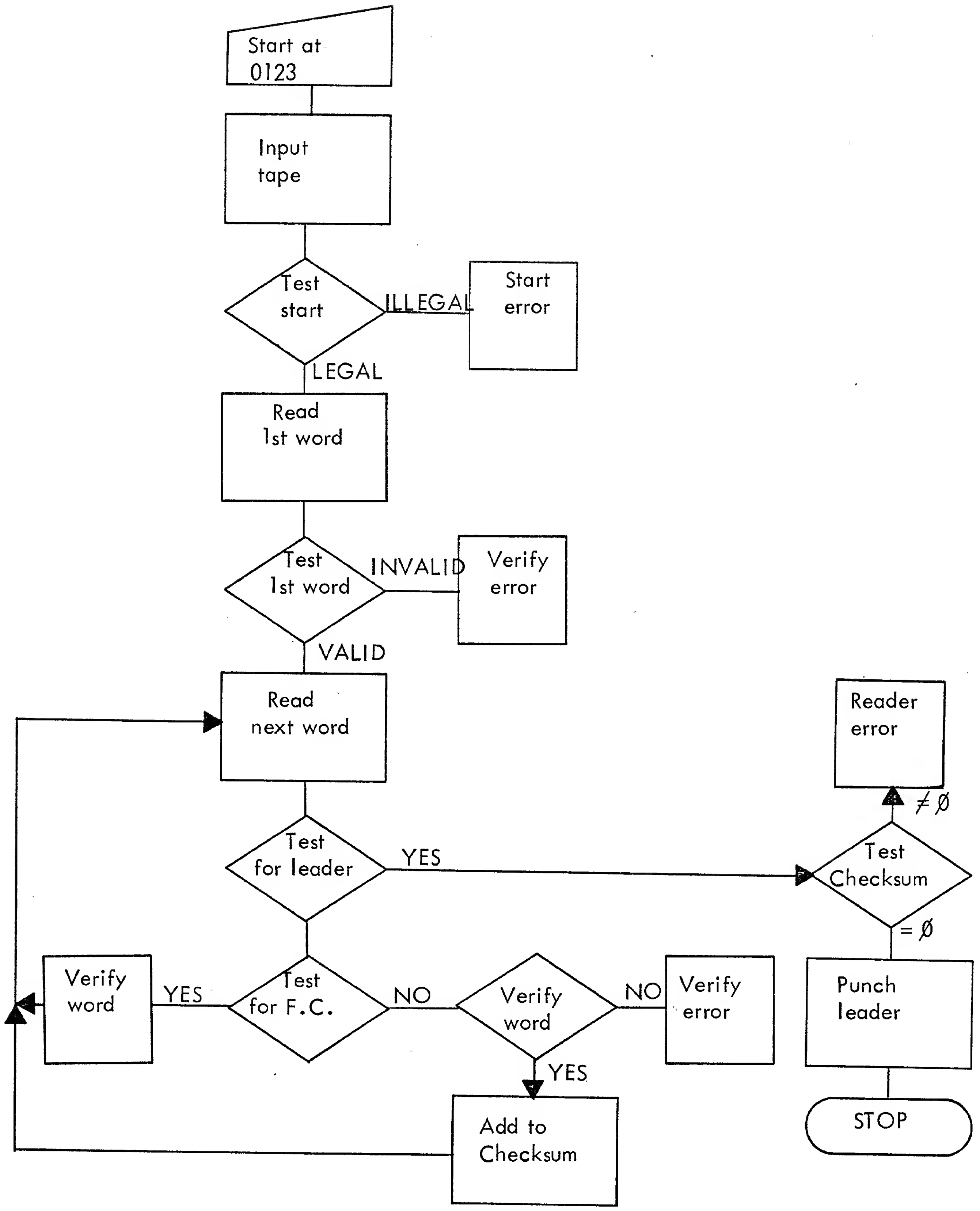
Not Applicable


A
Store pointers
Punch leader
Load data word
Test for origin address
NO
Test for F.C. Char.
YES
Output Character
YES
Output origin
NO
Output word
Test for last word
NO
YES
END OUTPUT

OUTPUT MODE


Start at 0001
Specify low or high speed peripherals
Set up buffer pointers
Read 1st word
Test for leader
YES
NO
Test for blank tape
YES
NO
START ERROR
Read next word
Test for leader
YES
NO
Get and Assemble 12-bit data words
Test for origin
YES
Store address
NO
Decrement address buffer pointer
Test for F.C. Char.
YES
Move origin add. buffer down 1
Store word in data buffer
Test for buffer overlap
YES
Overflow error
NO
Read next word
Test for leader
NO
YES
End of input
A

INPUT MODE


Start at 0123
Input tape
Test start
ILLEGAL
Start error
LEGAL
Read 1st word
Test 1st word
INVALID
Verify error
VALID
Read next word
Test for leader
YES
Reader error
≠ Ø
Test Checksum
= Ø
Verify word
YES
Test for F.C.
NO
Verify word
NO
Verify error
Punch leader
YES
Add to Checksum
STOP

VERIFY MODE

1. INTRODUCTION

A. PROGRAM SUMMARY

The Verifier-Reloader Program compares stored memory data with a paper tape and ferifies the contents. Any differences are printed on the teletype and can be reloaded or left unchanged.

B. PROGRAM AREA

$74ØØ_8$ through 7574_8.

C. STARTING ADDRESS

$74ØØ_8$.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor equipped with an ASR-33 Teletype. The optional peripheral is a high speed reader.

E. DEFINITIONS

None

2. PROGRAM DESCRIPTION

This program is of great value to the user in debugging. Questionable programs and overlays can be quickly verified and reloaded. Any differences between the stored memory data and the paper tape are read printed on the teletype. The following is an example of a verification output.

```
F1
1473            ØØ24          Ø241
FØ
F2
```

Each memory field is identified whether or not an error is encountered.

The memory address where the difference is encountered is printed first (1473), next the current content of this location (ØØ24), and third, the correct value (Ø241).

Switch Register BIT 1 set to "Ø" allows the user to reload the erraneous location from the paper tape and if set to "1" the current content is left unchanged. In either case, the program will print the location, current content and correct value.

At the completion of the verification run, the checksum is calculated and printed on the teletype.

3. OPERATOR OR USER CONTROL

Switch Register BIT Ø set to "Ø" specifies the high speed reader as the input device and set to "1" specifies the low speed reader (ASR-33 Teletype).

Switch Register BIT 1 set to "Ø" automatically reloads the erroneous location from the paper tape and if set to "1" the current content is left unchanged.

4. OPERATIONAL PROCEDURE

1) Load the Verifier-Reloader High/Low Speed program tape with the Binary Loader (refer to the Binary Loader (ND41-0005) for a detailed description of its use).

2) Set the Switch Register to $74ØØ_8$.

3) Depress LOAD AR.

4) Specify the input device and reload option if desired (BITS Ø and 1).

5) Place the verification program into the selected reader with the leader at the read station.

6) Depress START.

7) The program will now be read in and compared with the data stored in memory. Verification errors are read out and the correct values reloaded (if specified in Step 4). At completion, the checksum is printed on the teletype. Any value other than ØØØØ indicates a reader error. Re-start this procedure from Step 1.

5. ERROR DIAGNOSTICS

Any differences between the stored memory data and the paper tape are read out to the teletype. The memory field where the difference is encountered is printed first, next the address, the current content of this location, and the correct value.

6. COMMAND SUMMARY

```
Bit Ø = Ø - Read from teletype reader.
      = 1 - Read from HS reader.

Bit 1 = Ø - Reload program.
        1 - Leave memory unchanged.
```

7. FLOW CHART

Not Applicable.

8. PROGRAM LISTING

Not Applicable.

1. INTRODUCTION

A. PROGRAM SUMMARY

The Short Form Binary Loader is a single field loader occupying a minimum of memory that loads binary formatted program records into the ND812 Central Processor via the low speed paper tape reader.

B. PROGRAM AREA

76ØØ8 through 7647_8.

C. STARTING ADDRESS

76ØØ8.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor equipped with an ASR-33 Teletype. There are no optional devices.

E. DEFINITIONS

1. PROGRAM RECORD

The collection of binary words, etc., which when interpreted by the appropriate loader, results in the processor being loaded with machine instructions and data executable as a program. The Object Coding produced by the assembler is the commonest example of a program record. The Binary Loader discussed here is designed to interpret object coding from the BASC-12 Assembly Language Processor, program records produced by the Multiple Field Binary Writer, or any other records in " Binary Format".

2. BLOCK

A series of consecutive 12-bit memory addresses preceded by an origin. A program record consists of one or more blocks.

3. ORIGIN

A 12-bit word which is interpreted by the Binary Loader as an address where the first 12-bit word of a block is to be loaded. Succeeding words of the block are loaded into consecutive memory locations until another origin is detected or the end of the program record is reached. Thus each block is preceded by an "origin".

2. PROGRAM DESCRIPTION

Since the Binary Loader is itself a program, some means must be provided to load it into memory. Unlike the Paper Tape and Cassette Loader (ND41-0005) using a Bootstrap, this program must be manually loaded through the Switch Register. This loader can also be loaded with the Hardware Loader, if the processor is equipped with this option.

In loading a program from a paper tape reader, the Binary Loader begins by reading leader. Actual loading of the processor begins when the Binary Loader detects the first character different from $Ø2ØØ_8$ (eight level punch only). For this reason, it is essential that a program tape be placed in the reader with the leader at the read station. Should the program tape be placed in the reader with blank tape at the read station, the Binary Loader will begin loading zeros into memory beginning with location $ØØØØ_8$. Blank tape is not a 200 level and the loader assumes that it must begin loading another program. The actual leader of the program record will be interpreted as trailer.

The loading process consists of assembling consecutive pairs of frames using levels one through six as high and low order halves of 12-bit words. The assembled 12-bit words are stored in the processor in consecutive memory locations as determined by the presence and interpretation of the origins on the paper tape. When a program record is created, the last two characters are written in such a way as to make the sum of all the 12-bit words (including the last) on the tape equal to zero. The loader keeps a running sum or checksum of the 12-bit words in the record. When the loader detects trailer, it tests the checksum to determine whether or not the loading process has been correct. If the checksum is zero, the loading process is assumed to be correct and the loader will stop with the J register equal to zero. If the checksum is non-zero, the loading process may be assumed to be in error. In any case, the checksum is left in the J register at the completion of the loading process.

The following describes the various parts of Figure 1.

1) Leader/Trailer - Leader and trailer are punched identically and indicate the beginning and end of a binary format program respectively on paper tape. Program records written on magnetic tape substitute a file mark and tagword for the leader.

2) Origin - A 12-bit word punched in two frames of six bits each interpreted by the Binary Loader as an address at which to begin storing programs. Origins are generated on a program source tape with the use of the special character asterisk (*). An origin is distinguished from program words on the tape by the presence of the seven level punch. Levels one through six are the higher 6-bits of the address. The low order 6-bits may be found in levels one through six of the next tape frame.

3) Program - The program consists of 12-bit data words which are to be stored in the processor for execution as the user's program. The 12-bit data words appear on the tape as two consecutive frames of 6-bits each. The seven and eight levels must not be punched.

4) Checksum - This is a running sum of all the 12-bit words on the tape, including the program words and origin, excluding field change characters and the checksum itself. Thus, the sum of all the 12-bit words on the tape including the check-sum should be zero. If the result is not zero, the program was not correctly loaded and must be reloaded by repeating the operational steps of the Binary Loader. The checksum is always written in two halves as the last two frames in the record and is further indicated by a 7-level punch on the first half.

NOTE

Field Change characters will be ignored. The loader can load only into the field in which it is itself located.

3. OPERATOR OR USER CONTROL

None.

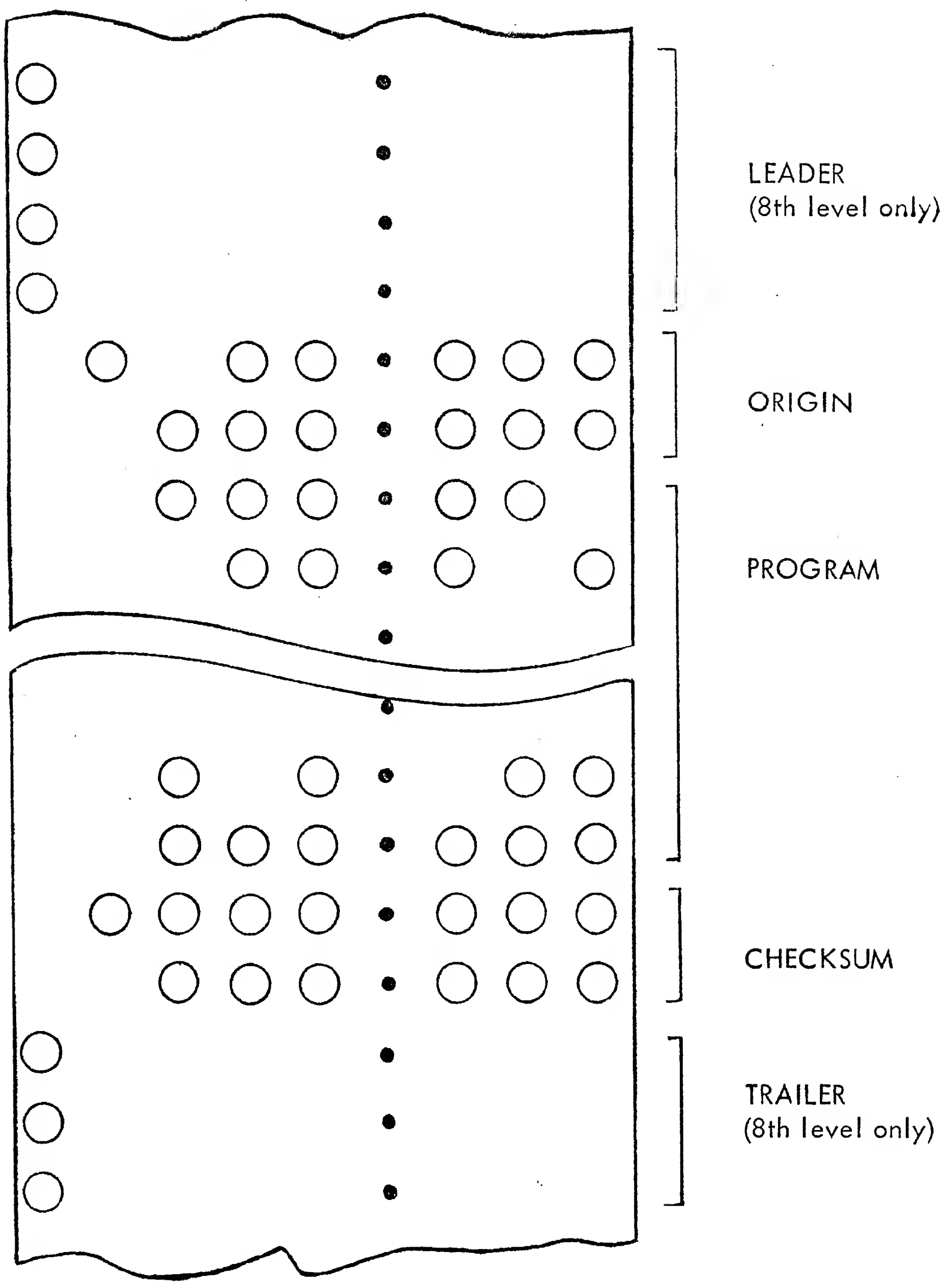

LEADER
(8th level only)
ORIGIN
PROGRAM
CHECKSUM
TRAILER
(8th level only)

FIGURE 1

4. OPERATIONAL PROCEDURE

The following is the procedure by which the Short Form Binary Loader is manually loaded into the processor.

1) Place the POWER ON/POWER OFF/CONTROL OFF switch in the POWER ON position.

2) Set the Switch Register to $76ØØ_8$ and depress LOAD AR.

3) It is now necessary to load forty instructions into the Switch Register, each of which is followed by depressing the LOAD MR switch.

ADDRESS	INSTRUCTION	ADDRESS	INSTRUCTION	ADDRESS	INSTRUCTION
76ØØ	74Ø3	762Ø	5Ø27	764Ø	61Ø5
76Ø1	1751	7621	ØØØØ	7641	15Ø1
76Ø2	5445	7622	1222	7642	631Ø
76Ø3	6427	7623	ØØØØ	7643	44Ø3
76Ø4	61Ø1	7624	55Ø1	7644	3512
76Ø5	1346	7625	4422	7645	6313
76Ø6	11Ø1	7626	5421	7646	Ø2ØØ
76Ø7	6423	7627	64Ø3	7647	ØØØØ
761Ø	6Ø1Ø	763Ø	611Ø		
7611	1122	7631	6124		
7612	1615	7632	ØØØØ		
7613	6Ø11	7633	74Ø4		
7614	56Ø7	7634	61Ø1		
7615	34Ø6	7635	74Ø3		
7616	14ØØ	7636	4Ø1Ø		
7617	6ØØ6	7637	15Ø7		

4) To check if the instructions were stored in the proper locations, set the Switch Register to 7762_8 and depress LOAD AR. Place the SELECT REGISTER switch in the ADDRESS position. Depressing the NEXT WORD key causes the Selected Register indicator lamps to display the current address and MEMORY REGISTER indicator lamps the contents of the address. Continue to depress the NEXT WORD key until all instructions have been checked.

The following is the Hardware Loader procedure:

1) Place the paper tape of the Short Form Binary Loader in the low speed reader (ASR-33 Teletype) taking care to place leader at the read station, and turn reader on.

2) Simultaneously depress both the LOAD ADDRESS and NEXT WORD switches.

3) The paper tape is read-in and will stop on trailer. If the J register equals zero, the loading process was correct. If the J register is non-zero, repeat from Step 1.

The Binary Loader is now in memory and is used to load other program records from paper tape.

LOW SPEED PAPER TAPE

1) Set the Switch Register to $76ØØ_8$ and depress LOAD AR.

2) Place the program tape to be read into the ASR-33 Reader with the leader at the read station.

3) Place the START/STOP switch on the ASR-33 Reader to START.

4) Depress START.

5) The Binary Loader will read the program tape and stop at trailer. If the content of the J register is zero at completion of the loading process, the program tape was loaded correctly. If the J register is non-zero, re-start this loading procedure from Step 2.

5. ERROR DIAGNOSTICS

The checksum is stored in the J register at the completion of a program loading procedure. A non-zero J register indicates an erroneous load. Refer to the section of the OPERATIONAL PROCEDURE and re-load the program.

6. COMMAND SUMMARY

None.

7. FLOW CHART

Not Applicable.

8. PROGRAM LISTING

Not Applicable.

1. INTRODUCTION

A. PROGRAM SUMMARY

The Binary Writer for High Speed Punch program is an independent Single Field Binary Writer designed to create binary formatted program records via a high speed punch while itself occupying a minimum of memory.

B. PROGRAM AREA

$64Ø Ø_8$ through 6476_8.

C. STARTING ADDRESS

$64ØØ_8$.

D. EQUIPMENT CONFIGURATION

Minimum requirements are a ND812 Central Processor equipped with a high speed paper tape punch.

E. DEFINITIONS

None.

2. PROGRAM DESCRIPTION

Switch Register controls allow the user to create program records from any arbitrary block of memory. These blocks are written on paper tape via the high speed punch and can consist of up to 7777_8 consecutive memory locations. (This program occupies locations $64ØØ_8$ through 6476_8.) This is a Single Field Binary Writer which limits the record selection to the field in which the Binary Writer is located.

After the Starting Address is loaded, the number of blocks to be written is loaded into the Switch Register and the program is started. The program will then punch a leader (8th level representing $2ØØ_8$) and stop. The Starting Address of the block is now set into the Switch Register, followed by depressing CONTINUE. The program will stop again to allow the user to enter the Stopping Address of the block. Depressing CONTINUE causes the program to output all data between and including the selected Starting and Stopping addresses. If more than one block was specified the program will stop. The user need only to enter the Starting and Stopping Addresses for each additional block. At the completion of the last block the checksum and trailer are written to close the record.

Since the output process is non-destructive, the user is free to output any portion of memory. However, care should be taken when outputting that portion of memory ($76ØØ_8$ through 7777_8) which will be occupied by the Binary Loader when the record is re-loaded. It is not possible for the Binary Loader to load itself.

If the Starting Address is larger than the Stopping Address, the Binary Writer will begin with the Starting Address and output consecutive memory locations to the end of memory. At this point the program will "wrap around" and continue outputting from location $ØØØØ_8$ until the Stopping Address is reached.

3. OPERATOR OR USER CONTROL

Only the number of blocks and the Starting and Stopping Addresses can be controlled by the user. In each case the entire 12-bit Switch Register word can be utilized to specify these controls.

4. OPERATIONAL PROCEDURE

1) Load the Binary Writer for High Speed Punch program tape with the Binary Loader (refer to the Binary Loader (ND41-0005) for a detailed description of its use).

2) Set the Switch Register to 64ØØ and depress LOAD AR.

3) Set the number of blocks to be written into the Switch Register.

4) Depress START.

5) The program will punch the leader and stop.

6) Set the Starting Address into the Switch Register.

7) Depress CONTINUE.

8) The program will now stop.

9) Set the Stopping Address into the Switch Register.

10) Depress CONTINUE.

11) The program will output the first block and stop.

12) If more than one block was specified, return to Step 6 and specify new Starting and Stopping Addresses.

13) At the completion of the last block, the program will write the checksum and trailer, then stop.

5. ERROR DIAGNOSTICS

None.

6. COMMAND SUMMARY

None.

7. FLOW CHART

Not Applicable.

8. PROGRAM LISTING

Not Applicable.

1. INTRODUCTION

A. PROGRAM SUMMARY

This program is an independent program that allows the user to examine and modify any word in a Memory Field.

B. PROGRAM AREA

$6\emptyset\emptyset\emptyset_8$ through 6252_8.

C. STARTING ADDRESS

$6\emptyset\emptyset\emptyset_8$.

D. EQUIPMENT CONFIGURATION

Minimum requirements are a 4K ND812 and an ASR-33 Teletype.

E. DEFINITIONS

N - Increments current or base address by one and closes current address.

↑ - Uses the last 7 bits of the current address contents to determine the effective address and closes the current address.

← - Uses the current address contents as absolute address and closes the current address.

- Returns to the base address and closes the current address.

SPACE - Prints contents of current address.

CARRIAGE-RETURN - The argument preceding the carriage return is deposited in the currently opened address. If no argument is given, the current address is closed.

M - Location in which the users mask is stored.

N - Upper and lower word search limits.

W - Search word.

2. PROGRAM DESCRIPTION

This program is controlled by special characters which are entered via the teletype keyboard. These characters permit the user to examine the contents of a particular address and check the results with the program listing. If the results are incorrect, the user can alter the location.

The entry address is stored in a base and current address software registers. The base address is updated by a new entry address or incremented with the control character "N". The " ↑ " and "←" controls will not affect the base address and after initiating one of these controls, the base address can no longer be incremented. The current address reflects the contents of the base address until either a " ↑ " or a "←" is used. These controls will branch the program and together with the "N" control, update only the current address. The control character "#" returns the program to the base address.

A word search routine is incorporated in this program that prints on the teletype the address of any preselected octal number. The search operation is the logical "AND" of the user mask (M) and the binary words within the set limits (N).

3. OPERATOR OR USER CONTROL

```
7ØØØ        4321        1234 ↑
7Ø34        4242        2ØØØ ←
2ØØØ        3167        6543#
7ØØØ        1234
```

In the above example $7ØØØ_8$ was typed on the teletype keyboard and is considered to be the entry address. Next the SPACE bar is depressed and the contents of the entry address (4321_8) is printed. The argument 1234_8 is given preceding a " ↑ ".

The " ↑ " will close $7ØØØ_8$ (now containing 1234_8) and use the last 7 bits of the argument to determine the effective address. In this example the current address is advanced 34_8 locations, if the argument was 1334_8 the current address would be diminished 34_8 locations. Location $7Ø34_8$ is now opened, the contents printed and the argument $2ØØØ_8$ is given preceding a "←". Typing "←" will close $7Ø34_8$ (now containing $2ØØØ_8$) and use $2ØØØ_8$

as the absolute address. Location $2ØØØ_8$ is opened, the contents printed and the argument 6543_8 is given preceding a "#". Typing "#" will close $2ØØØ_8$ (now containing 6543_8) and return the program to the base address ($7ØØØ_8$).

```
M7677         7777          N
6247          ØØØØ          N
625Ø          7777          7ØØØ
137W
ØØ27          1374
16Ø3          1374
1645          1374
61Ø1          1374
6174          1374
62Ø2          1374
6217          1374
7ØØØ          1374
```

In the above example the mask location was opened by typing "M" and the contents printed. The argument 7777_8 was given preceding a "N". Typing "N" will close the mask (now containing 7777_8) and open the lower search limit address (6247_8). The contents of the lower search limit is printed and no argument is given preceding the "N". Typing "N" for the second time closes the unchanged lower search limit and opens the upper search limit. The contents of the upper search limit is printed and the argument $7ØØØ_8$ is given. Next the CARRIAGE RETURN is depressed closing the lower search limit (now set to $7ØØØ_8$). 1374_8 is entered as the word search number and "W" is typed. The logical "AND" of the users mask (7777_8) and the binary words within the set limits is compared with the search word and if equal, the address is printed.

```
M7777         Ø             N
6247          ØØØØ          1ØØØ          N
625Ø          7777          2ØØØ
ØW
1ØØØ          1738
.             .
.             .
.             .
.             .
.             .
2ØØØ          ØØ17
```

The above example, the word search routine is used to print a list from location $1ØØØ_8$ through $2ØØØ_8$. This is accomplished by setting the mask to Ø, the limits to $1ØØØ_8$ and $2ØØØ_8$, and setting the search word to Ø.

4. OPERATIONAL PROCEDURE

1) Load program tape with Binary Loader (refer to Binary Loader procedure).

2) Ascertain that the program to be debugged is in core.

3) Set Switch Register to $6ØØØ_8$.

4) Depress LOAD ADDRESS.

5) Depress START.

6) Input entry address and refer to OPERATOR OR USER CONTROL Section for a detailed description of control characters.

5. ERROR DIAGNOSTICS

Entering an illegal character or erroneous number will close the current address, cause the Teletype to print a "?" and return the carriage.

6. COMMAND SUMMARY

None

7. FLOW CHART

Not Applicable.

8. PROGRAM LISTING

(Next Page)

1. INTRODUCTION

A. PROGRAM SUMMARY

This program is an independent Multiple Memory Field Binary Writer designed to create binary formatted program records via a high/low speed paper tape punch or magnetic tape cassette unit.

B. PROGRAM AREA

$74\not{0}\not{0}_8$ through 7567_8.

C. STARTING ADDRESS

$74\not{0}\not{0}_8$.

D. EQUIPMENT CONFIGURATION

Minimun requirements are a ND812 Central Processor equipped with a low-speed punch. Optional peripherals include the high speed punch or magnetic tape cassette unit.

E. DEFINITIONS

Initialization word - specifies device, tagword, and number of blocks to be written. It is normally acquired by the program from the Switch Register.

2. PROGRAM DESCRIPTION

This program is controlled via the Switch Register and allows the user to create program records from up to 15 (17_8) arbitrary blocks of memory. These blocks may be written on paper tape or magnetic tape cassettes and can consists of up to 4096 consecutive memory locations from any single memory field. An "overlap" of two memory fields in one block is not permissable. Different blocks may come from different memory fields.

After the Initialization word (See Chapter 3) has been set into the Switch Register, the program can be started. The program will then punch a leader (8th level representing 128_8 or 200_{10}) or write a file mark and tagword and stop. The Memory Field from which the first block is to be taken should now be set into the Switch Register (BITS 1Ø and 11). If BIT Ø is set to "1", the field change character preceding each data block will be suppressed. BITS 1 through 9 are ignored at this time. After the Memory Field number is set into the Switch Register, the user should depress the CONTINUE switch. The program will stop again to allow the user to enter the Starting Address of the block into the Switch Register, followed by depressing CONTINUE. The program will stop again to allow the user to enter the Stopping Address of the block. Depressing CONTINUE causes the program to output all data between and including the selected Starting and Stopping Addresses. If more than one block was specified (BITS 2-5 of the Initialization word) the program will stop. The user need only to enter the Memory Field number, Starting and Stopping Addresses for each additional block. At the completion of the last block the checksum and trailer are written to close the record.

Since the output process is non-destructive, the user is free to output any portion of memory. However, care should be taken when outputting that portion of memory ($76ØØ_8$ through 7777_8) which will be occupied by the Binary Loader when the record is re-loaded. It is not possible for the Binary Loader to load itself.

If an attempt is made to write on the end of the tape, the program will stop with the J register equal to 7777_8. The program should be re-started with a new cassette that is adequate to contain the entire block. Pressing CONTINUE at this point will not have any effect. Positioning the tape in a cassette is the user's responsibility as the program will not skip over previously written records.

If the Starting Address is larger than the Stopping Address, the Binary Writer will begin with the Starting Address and output consecutive memory locations to the end of memory. At this point the program will "wrap around" and continue outputting from location $ØØØØ_8$ on until the Stopping Address is reached.

3. OPERATOR OR USER CONTROL

The Initialization word is a 12-bit word used to specify the output device, number of blocks to be written, and tagword. The Initialization word Switch Register settings are as follows:

OUTPUT DEVICE SELECTION	BIT Ø	BIT 1
Cassette (center drive No. 2)	Ø	Ø
High Speed Paper Tape Punch	Ø	1
Low Speed Paper Tape Punch (ASR33)	1	1

BITS 6 through 11 are used for the tagword selection when writing on magnetic tape cassette. They are written on tape at the beginning of the program record and are used to identify the record to the Binary Loader.

BITS 2 through 5 indicate the number of blocks to be written, regardless of the output device selected. There may be up to $15_{1Ø}$ (17_8) blocks selected.

NOTE

> If BITS 2 through 5 are all "Ø's", the user will have selected 4096 blocks.

The Memory Field from which the first block is to be written is specified by BITS 1Ø and 11 via the Switch Register. Permissable Memory Field selections are Ø, 1, 2 and 3 (in 4K increments). If BIT Ø is set to "1", the field change character is suppressed.

The Starting and Stopping Addresses of a block are selected by the entire 12-bit Switch Register word.

4. OPERATIONAL PROCEDURE

1) Load the Binary Writer program tape with the Binary Loader (refer to the Binary Loader (ND41-0005) for a detailed description of its use).
2) Set the Switch Register to $74ØØ_8$ and depress LOAD AR.
3) Set the Initialization word into the Switch Register.
4) Depress START.
5) The program will stop after punching leader.
6) Set the Memory Field from which the block is to be written into the Switch Register (BITS 1Ø, 11 and Ø).
7) Depress CONTINUE.
8) The program will now stop.
9) Set the Starting Address into the Switch Register.
10) Depress CONTINUE

11) The program will now stop.

12) Set the Stopping Address into the Switch Register.

13) Depress CONTINUE.

14) The program will output the first block and stop.

15) If more than one block was specified by the Initialization word, return to step 6 and specify new parameters.

16) At the completion of the last block, the program will write the checksum and trailer, then stop.

5. ERROR DIAGNOSTICS

If an attempt is made to output to a non-existant device, the program will enter an endless loop.

If an attempt is made to write on the end of magnetic tape, the program will stop and the J register will equal 7777_8. The program should be re-started with a new cassette that is adequate to contain the entire block.

6. COMMAND SUMMARY

INITIALIZATION WORD

OUTPUT DEVICE	BIT Ø	BIT 1
Cassette (center drive No. 2)	Ø	Ø
High Speed Paper Tape Punch	Ø	1
Low Speed Paper Tape Punch (ASR33)	1	1

BITS 2 through 5 indicate the number of blocks to be written, with a maximum of $15_{1Ø}$ (17_8) blocks.

BITS 6 through 11 are used for the tagword selection when writing on magnetic tape cassette.

NOTE

If BITS 2 through 5 are all "Ø's", the user will have selected 4096 blocks.

The Memory Field from which the first block is to be written is specified by BITS 1Ø and 11 via the Switch Register. Permissable Memory Field selections are Ø, 1, 2 and 3. Bit Ø set to "1" suppresses the field change character.

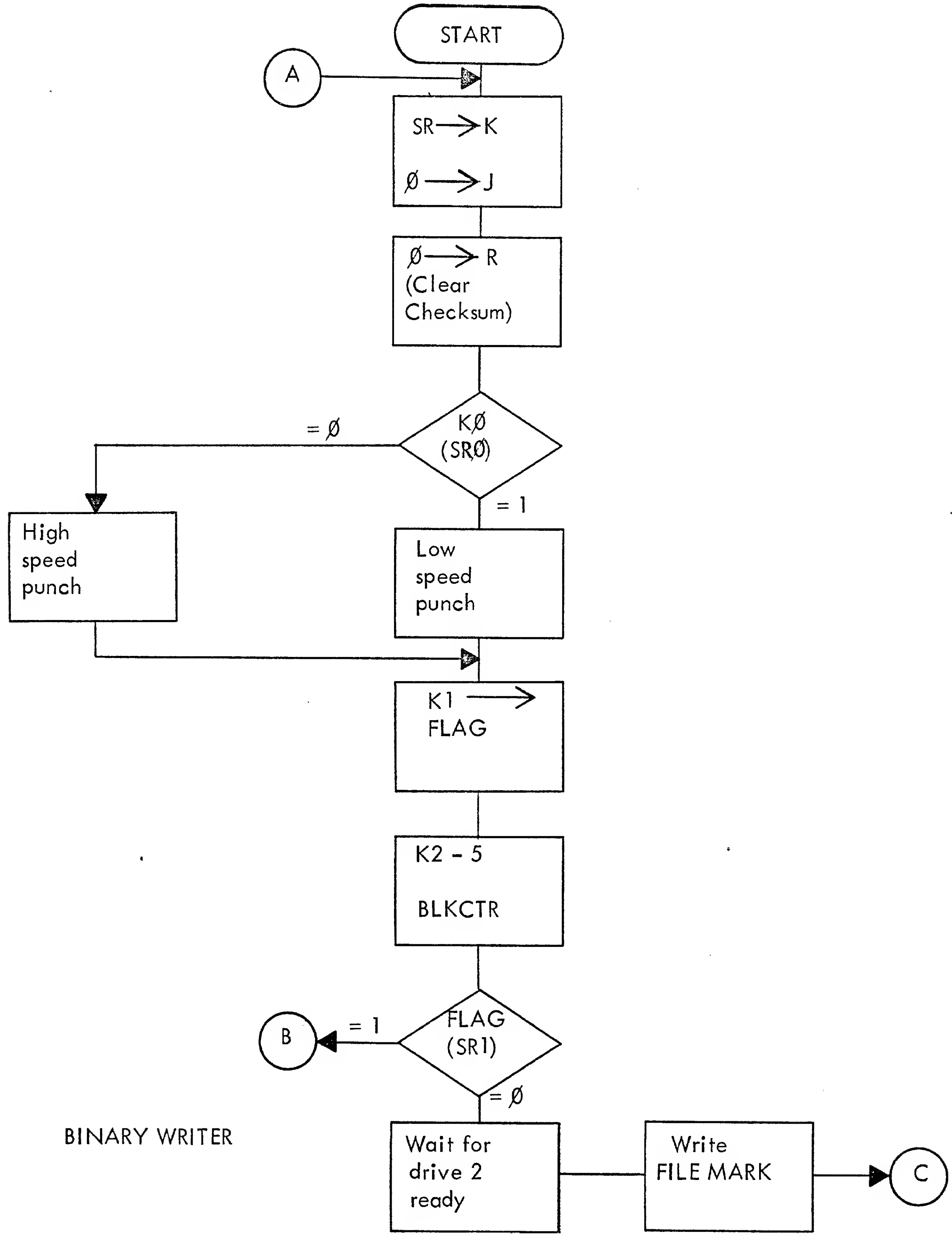
The Starting and Stopping Addresses of a block are selected by the entire 12-bit Switch Register word.

7. FLOW CHART

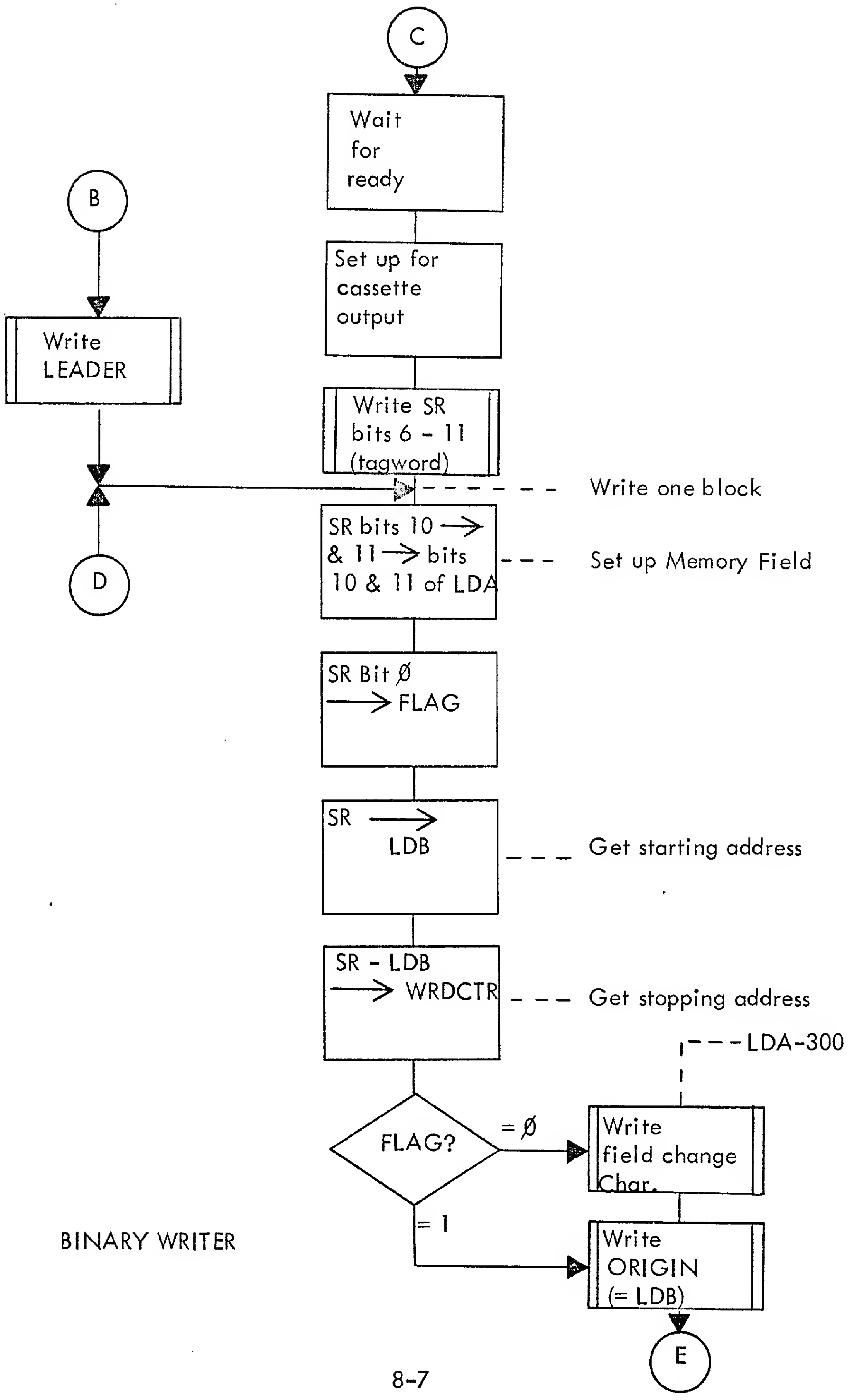
(Next Page)

8. PROGRAM LISTING

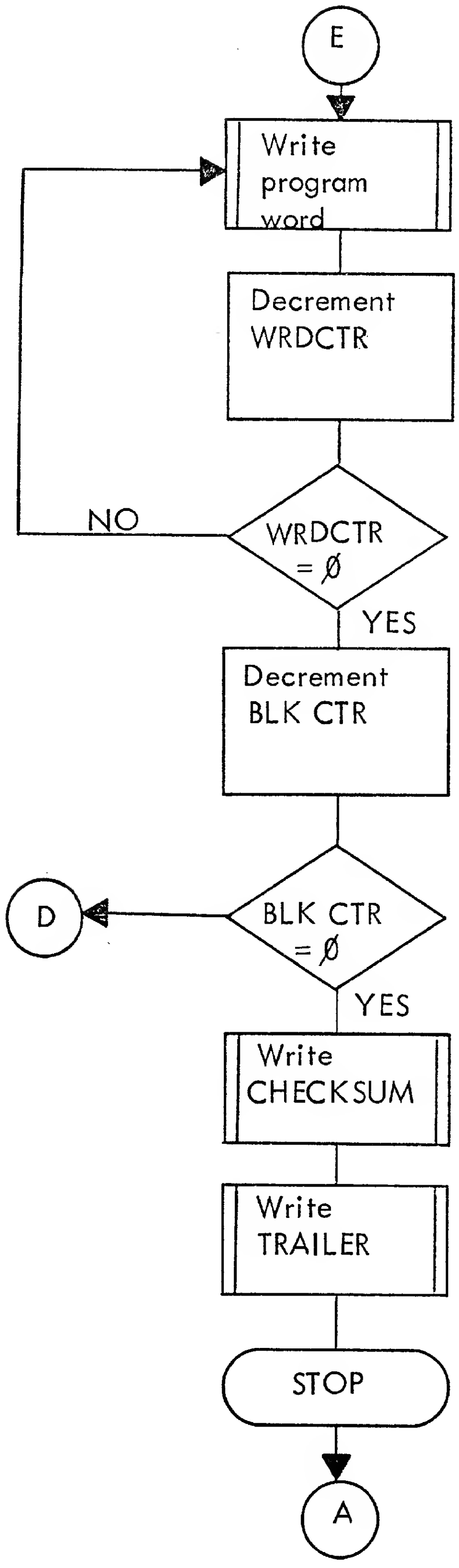
Not Applicable.



BINARY WRITER



BINARY WRITER



BINARY WRITER

1. INTRODUCTION

A. PROGRAM SUMMARY

The Floating Point Program is a multiple field arithmetic floating point and I/O package.

B. PROGRAM AREA

The program occupies location $Ø4ØØ_8$ through 2751_8 and uses location 2752_8 through 2761_8 as a buffer.

C. STARTING ADDRESS

The routines are called by software commands from the main program.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor equipped with an ASR-33 Teletype.

E. DEFINITIONS

None

2. PROGRAM DESCRIPTION

The Floating Point Program is an arithmetic and basic I/O package that allows the user to prepare programs without concern for the decimal point. The significant digits are maintained for each number, enhancing the accuracy of any series of calculations.

Floating Point Arithmetic is of special interest in those cases where many multiplications and divisions are performed and the magnitudes are likely to differ greatly. The ability to store both large and small numbers by storing the significant digits and an exponent is the primary advantage of a floating point system.

The Floating Point Program is written as a self-contained package with its own I/O and arithmetic routines, so the programmer need not prepare his own packages.

A floating point number consists of a number (mantissa) times a base raised to a power (exponent). For example, in the decimal system, 15 may be expressed in many ways:

1500.0	$*10^{-2}$
150.00	$*10^{-1}$
15.000	$*10^{0}$
1.5000	$*10^{1}$
.15000	$*10^{2}$

etc.

All floating point numbers are stored internally as binary numbers. For example, 7 in decimal is 111 in binary and may be written as:

11100.0	$*2^{-2}$	(28 * 1/4 = 7)
1110.00	$*2^{-1}$	(14 * 1/2 = 7)
111.000	$*2^{0}$	(7 * 1 = 7)
11.1000	$*2^{1}$	(7/2 * 2 = 7)
1.11000	$*2^{2}$	(7/4 * 4 = 7)

etc.

The binary exponent is always a signed integer. A number is considered to be normalized when a value is expressed:

MANTISSA * 2^E

Where the Mantissa is a signed quantity. More significant bits are stored as a result of this convention. For example, .1 (Decimal) written in an unnormalized binary format is:

Ø.ØØ Ø11 ØØ1 1ØØ 11Ø Ø11 ØØ1 1ØØ 11Ø Ø11

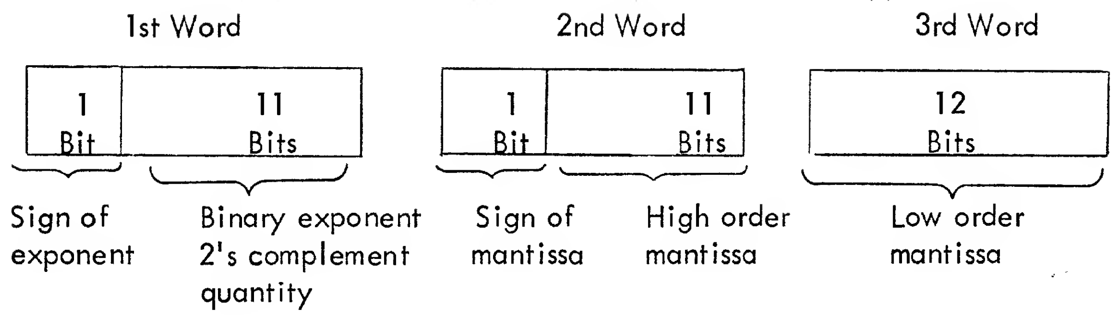
If stored in 24 bits, the leading Ø's are not significant and only 20 meaningful bits are maintained. By rewriting and normalizing the number as:

2^{-3} * Ø.11 ØØ1 1ØØ 11Ø Ø11 ØØ1 1ØØ Ø11 ØØ1 1ØØ . . .

23 bits of significance can be stored.

A number is considered to be normalized when the absolute value of the mantissa is greater than 1/2 and less than 1. The binary point falls after the first bit (Ø. indicating a positive number 1. a negative number).

The form of a floating point number consists of three 12 bit words. They are:



Thus .1 (decimal) would be stored as:

```
111   111   111   101
011   001   100   110
011   001   100   011
```

or in octal:

```
7775
3146
3143
```

The address of this number is considered to be that of the exponent; that is, if the exponent has address 3210.

3210/7775
3211/3146
3212/3143

Then the number's address is 3210. The number -.1 (decimal) is equal to .063145314 (octal) or ØØØ 11Ø Ø11 ØØ1 1ØØ 11Ø Ø11 ØØ1 1ØØ (binary).

Upon normalization this becomes:

$(2^{-3})\ \underline{*}\ (\text{Ø.11 ØØ1 1ØØ 11Ø Ø11 ØØ1 1ØØ 11Ø})_2.$

In 2's complement, this is:

$(2^{-3}) * (\text{1.ØØ 11Ø Ø11 ØØ1 1ØØ 11Ø Ø11 ØØ1})_2.$

which is:

$(2^{-3}) * (4631\ 4632)_8$

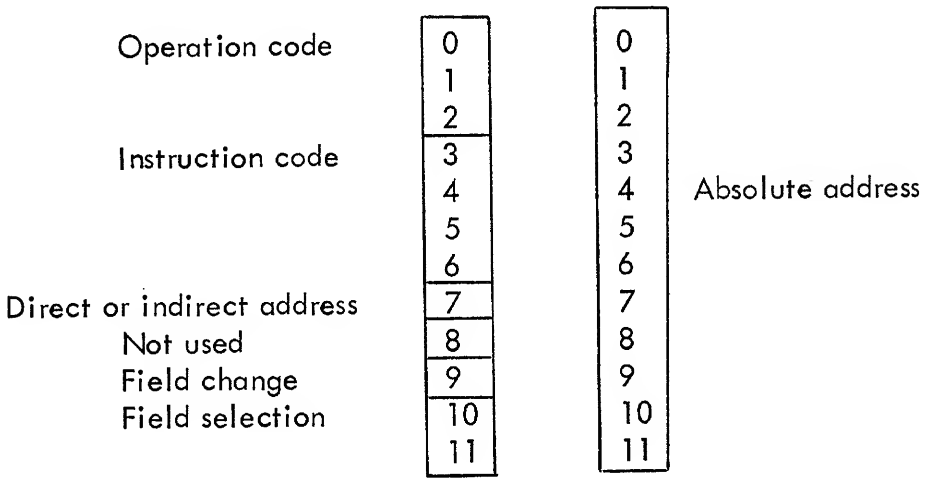
or:

7775
4631
4632

This number is normalized with the binary point between BITS Ø and 1 of the high order word.

All operations are called through the Floating Point Interpreter which is entered with a TWJPS to FPNT (Ø4ØØ8).

The standard Two-word Memory Reference Instruction Code format is used:



The basic Floating Point Package Commands are either two word or single word instructions that increment a single pointer by three, increment or decrement if the counter is zero, provide a floating jump, store and load the FACC and provide four arithmetic subroutines. A brief description of the arithmetic provided four arithmetic subroutines. A brief description of the arithmetic routines are as follows:

Floating subtraction is performed by negating the operand and then calling the addition routine. The result is normalized and control is returned to the interpreter.

Floating addition is performed by aligning the exponents of the numbers and then adding the mantissas. Both numbers are shifted one bit to the right, preventing overflow into the sign bit. A 2's complement addition is performed, the result is normalized, and control is returned to the interpreter.

Floating division is performed by subtracting the divisor exponent from the dividend exponent, dividing the mantissas and normalizing the result. Control then returns to the interpreter.

Seven additional instructions are available with this basic package as single word instruction, they are; input and output routines, three operates (halt computer, exits floating package, and normalize FACC) and two functions that will convert a floating point number to a double precision integer and float a double precision integer.

All Floating Point Commands are multiple field, except the Floating Jump (FJMP). The user must exit the floating point package in order to perform a jump to another field.

All Floating Point Operations are performed through a Floating Accumulator (FACC). FACC occupies core locations Ø537$_8$, Ø54Ø$_8$ and Ø541$_8$. Location Ø537$_8$ (EXPON) contains the exponent and sign, location Ø54Ø$_8$ (HORD) contains the high order 11 mantissa bits and sign, and location Ø541$_8$ (LORD) contains the low order 12 mantissa bits.

3. OPERATOR OR USER CONTROL

FLOATING POINT INPUT

The input routine reads decimal characters from the ASR-33 keyboard which are internally converted to the Binary Floating Point format. There are several acceptable forms for entering numbers, e.g., the number 583.9 may be entered in any of the following ways:

583.9
.5839E3
.5838E+03
+5839E-1
Etc.

Input is halted by typing a character that is not a part of this format. The conversion of "14.0" would end at the second "." and the binary number

```
000  000  000  100          (octal 0004)
011  100  000  000          (octal 3400)
000  000  000  000          (octal 0000)
```

would be in the floating accumulator.

The routine is entered by an effective JPS 1661. Control is returned to the next statement after the JPS when the input conversion is terminated. The number is stored in the FACC (floating accumulator) in normalized floating binary format.

INPUT FLAGS

Two locations (flags) are associated with the input routine that can aid the user.

2166 - location 2166 contains the last character entered.

1756 - if location 1756 is zero, no conversion was made. A terminator was initially typed.

If a "RUBOUT" is entered before a terminating character, the input routine is restarted and the previous input is destroyed.

FLOATING POINT OUTPUT

The basic output form is the exponential, or E format. As an example, if FACC contained:

0003
2000
0000
+0.4000000E+01 would be printed.

The output may be formatted to print an integer or decimal notation. Three locations (2301, 2422 and 2600) control the output format. Location 2301 contains the number of digits to be outputted. If location 2301 is zero, the output will be in E format. Location 2472 contains the number of digits outputted to the right of the decimal place (does not apply to E format). If location 2472 is zero a decimal point will not be printed. Location 2600, when non-zero, supresses all leading spaces (does not apply to E format).

If the number to be outputted is larger than the specified field width (2301), "X's" will be printed in place of a number. Any number printed will be preceded with a sign and all leading zeros are suppressed.

The following example illustrates the Floating Point output formatting features. FACC contains 786.324 decimal.

2301	2472	2600	Output
0	1	1	+0.786324E+03
2	0	1	+XX
3	2	0	+786
6	2	0	+ 786.32
6	2	1	+786.32
5	2	0	+786.32
4	0	0	+ 786
4	0	1	+786

Also, location 2612 can be set non-zero and a carriage return/line feed will not be printed after a number is printed. This location is initially set to zero and a carriage return/line feed is printed.

The contents of FACC is not destroyed during output. The output routine is called by an effective JPS 2227. Control is returned to the statement following the JPS.

As an example of how the Floating Point Package may be used, consider the polynomial:

Y = (B*C)/(A**3) 6 $\underline{D}$ + E

Assuming all values are in core, this polynomial may be evaluated as follows:

```
START, TWJPS                    /ENTER FLOATING INTERPRETER
       FLOAD  A
       FMULT  A
       FMULT  A
       FSTOR  TEMP1             /A**3
       FLOAD B
       FMULT  C                 /B*C
       FDIV   TEMP1             /(B*C)/(A*3)
       FSUB   D                 /(B*C)/(A**3) - D
       FADD   E                 /(B*C)/(A**3) - D + E
       FSTOR  Y
       FOUTPT                   /OUTPUT Y
       FEXIT
       STOP
       $

       TEMP1, Ø
              Ø
              Ø
       TEMP2, Ø
              Ø
              Ø
```

4. OPERATIONAL PROCEDURE

1) Load the Floating Point program tape with the Binary Loader (refer to the Binary Loader (ND41-0005) for a detailed description of its use).

2) The routines are programmable subroutines called by the main program.

5. ERROR DIAGNOSTICS

None

6. COMMAND SUMMARY

SINGLE WORD COMMANDS

OCTAL CODE	MNEMONIC	DESCRIPTION
1400	FINPUT	Floating input.
2000	FOUTPT	Floating output.
2400	FPTR	Increments single word pointer by three.
3000	FDSZ	Decrements single word pointer by one and skips the next floating instruction when memory is equal to zero.
3400	FISZ	Increments single word pointer by one and skips the next floating instruction when memory is equal to zero.
4000	FSUB	Floating subtract.
4400	FADD	Floating addition.
5000	FLOAD	Floating load FACC.
5400	FSTOR	Floating store FACC.
6000	FJMP	Floating jump.
6400	FDIV	Floating divide.
7000	FMULT	Floating multiply

TWO WORD COMMANDS

OCTAL CODE	MNEMONIC	DESCRIPTION
0240	FTWPT	Increments single word pointer by three.
0300	FTWDS	Decrements single word pointer by one and skips the next floating instruction when memory is equal to zero.
0340	FTWIS	Increments single word pointer by one and skips the next floating instruction when memory is equal to zero.
0400	FTWSB	Two word subtract.
0440	FTWAD	Two word addition.
0500	FTWLD	Two word load FACC.
0540	FTWST	Two word store FACC.
0600	FTWJP	Two word jump.
0640	FTWDV	Two word divide.
0700	FTWMT	Two word multiply.

OPERATE COMMANDS

OCTAL CODE	MNEMONIC	DESCRIPTION
1000	FSTOP	Halts Computer
1001	FEXIT	Exits Floating Package
7405	IFIX	Fix Floating Point number in FACC (Double Precision)
7406	FLOAT	Float Double Precision Integer

7. FLOW CHART

Not Applicable.

8. PROGRAM LISTING

Not Applicable.

1. INTRODUCTION

A. PROGRAM SUMMARY

This program is an overlay for the Floating Point Program (ND41-0041) that provides an exponent, log, square, and square root routines.

B. PROGRAM AREA

The program occupies locations $3ØØØ_8$ through 3373_8 and updates interpreter locations $Ø624_8$, $Ø625_8$, $Ø632_8$ and $Ø634_8$.

C. STARTING ADDRESS

These routines user callable subroutines.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor equipped with an ASR-33 Teletype.

E. DEFINITIONS

None

2. PROGRAM DESCRIPTION

LOG

This routine calculates the natural logarithm of the absolute value of the number contained in FACC and stores the result in FACC.

The following identities are used in the logarithm routine:

For $X \geq 1$
$\text{Log}(X) = \text{Log}(2^M * A) \quad 1 < A < 2$
$= M * \text{Log}(2) + \text{Log}(A)$
$\text{LOG}(1 + X) = \sum_{i=1}^{8}$

where

C_1	$=+.9999964239$
C_2	$=-.4998741238$
C_3	$=+.3317990258$
C_4	$=+.2407338084$
C_5	$=+.1676540711$
C_6	$=-.0953293897$
C_7	$=+.0360884937$
C_8	$=-.0064553442$

For $0 < X < 1$
$\text{Log}(X) = -\text{Log}(1/X)$

EXPONENT

This routine raises E to the FACC power and stores the result in FACC.

The following identity is used in the exponent routine.

$$E^X = 2^X * \text{Log } 2\ E = 2^N + F = 2^N * 2F$$

Where N is an integer and $0 < F < 1$

$$2^F = 1 + \cfrac{2F}{A - F + B\,F^2 - \cfrac{C}{D + F^2}}$$

Where

A	= 9.95459578
B	= 0.03465735903
C	= 617.97226053
D	= 87.417497202
Log_2 e	= 1.4426950409

If $X < \emptyset$

$e^X = 1/e-X$

SQUARE ROOT

This routine calculates the square root of the number contained in FACC and stores the result in FACC.

The square root is calculated by using Newton's method.

$X_i + 1 = 1/2\ (X_i + N/X_i)$

For the square root of N.

SQUARE

This routine calculates the square of the number contained in FACC and stores the result in FACC.

3. OPERATOR OR USER CONTROL

These routines are programmable subroutines.

4. OPERATIONAL PROCEDURE

1) Load program tape with the Binary Loader (refer to Binary Loader (ND41-0005) for detailed description of its use).

2) These routines are programmable subroutines and are called by the main program.

5. ERROR DIAGNOSTICS

None

6. COMMAND SUMMARY

OCTAL CODE	MNEMONIC	DESCRIPTION
7401	FSQRT	Square root of FACC.

OCTAL CODE	MNEMONIC	DESCRIPTION
7402	FSQUAR	Square FACC.
7407	FLOG	Natural log of FACC.
7411	FEXP	Exponent of FACC.

7. FLOW CHART

Not Applicable.

8. PROGRAM LISTING

Not Applicable.

1. INTRODUCTION

A. PROGRAM SUMMARY

This program is an overlay for the Floating Point Program (ND41-0041) that provides a sine, cosine, and ARC tangent routines.

B. PROGRAM AREA

The program occupies locations $34ØØ_8$ through $37Ø1_8$ and updates interpreter locations $Ø626_8$, $Ø627_8$ and $Ø633_8$.

C. STARTING ADDRESS

These routines are user callable subroutines.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor equipped with an ASR-33 Teletype.

E. DEFINITIONS

None

2. PROGRAM DESCRIPTION

SINE

This routine calculates the sine of the angle contained in FACC (assumed to be in radians) and stores the result in FACC.

The following identities are used in the sine routine:

$\mathrm{SIN}(-X) = -\mathrm{SIN}(X)$
$\mathrm{SIN}(X) = \mathrm{SIN}(2M + A) = \mathrm{SIN}(A)$
$\mathrm{SIN}(\pi - A). = -\mathrm{SIN}(A)$

Where $0 < A < 2\pi$ and M is an integer. With these identities, the argument a is reduced to the range $-\pi/2 < A < \pi/2$. If $X = 2A/\pi$, $-1 < A < 1$, then $\mathrm{SIN}(X) = A + C_3 A^3 + C_5 + C_7 A^7 + C_9 A^9$.

where

C1 = 1.57079631847
C3 = .64596371106
C5 =+.07968967928
C7 = -.00467376557
C9 =+.00015148419

The sine routine is valid over the range $-2\pi * .2047 < X < 2\pi * .2047$.

COSINE

This routine calculates the cosine of the angle contained in FACC (assumed to be in radians) and stores the results in FACC.

The cosine routine uses the identify $\mathrm{COS}(X) = \mathrm{SIN}(\pi/2 - X)$ and then uses the sine **routine.**

ARC TANGENT

This routine calculates the ARC tangent of the number contained in FACC and stores the result, in radians, in FACC.

The following identities are used in the ARC tangent routine:

$\mathrm{ATAN}(-X) = -\mathrm{ATEN}(X)$
IF $X > 1$
$\mathrm{ATAN}(X) = \pi/2 - \mathrm{ATAN}(1/X)$
For $0 < X < 1$

$$\mathrm{ATAN}(X) = \frac{X(A_0 + A, X^2 + A_a X^4)}{B_0 + B_1 X^2 + B_2 X^4}$$

where

$A_0 =$	.6402491953
$A_1 =$	.4229908144
$A_2 =$	.0264694361
$B_0 =$	.6402487022
$B_1 =$	.6363779373
$B_2 =$	.1108328778

with results in the range:

$-\pi/a < \text{ATAN}(X) < \pi/2$
$-00 < X < 00$

3. OPERATOR OR USER CONTROL

These routines are programmable subroutines.

4. OPERATIONAL PROCEDURE

1) Load program tape with the Binary Loader (refer to Binary Loader (ND41-0005) for detailed description of its use).

2) These routines are programmable subroutines and are called from the main program.

5. ERROR DIAGNOSTICS

None.

6. COMMAND SUMMARY

OCTAL CODE	MNEMONIC	DESCRIPTION
7403	FSIN	Sine of FACC.

OCTAL CODE	MNEMONIC	DESCRIPTION
7404	FCOS	Cosine of FACC.
7410	FATAN	ARC tangent of FACC.

7. FLOW CHART

Not Applicable.

8. PROGRAM LISTING

Not Applicable.

1. INTRODUCTION

A. PROGRAM SUMMARY

This program is an overlay for the Floating Point Program (ND41-0041) that provides Floating Point Operate (FNEG, FCLR, FINC, FSIM, FSIP and FSIZ) Instruction.

B. PROGRAM AREA

The program occupies locations $Ø3ØØ_8$ through $Ø354_8$ and updates interpreter locations $Ø6ØØ_8$ through $Ø6Ø5_8$.

C. STARTING ADDRESS

These commands are programmable subroutines.

D. EQUIPMENT CONFIGURATION

Minimum requirements are an ND812 Central Processor equipped with an ASR-33 Teletype.

E. DEFINITIONS

None

2. PROGRAM DESCRIPTION

The Operate Instructions augment the Floating Point Program allowing the user to execute a 2's complement of the floating accumulator (FNEG), FCLR clears and FINC increments the floating accumulator. Three skips are also included with the Operate Instructions that skip if the floating accumulator is negative (FSIM), positive (FSIP), or zero (FSIZ).

3. OPERATOR OR USER CONTROL

None.

4. OPERATIONAL PROCEDURE

1) Load program tape with Binary Loader (refer to Binary Loader (ND41-0005) for detailed description of its use).

5. ERROR DIAGNOSTICS

None.

6. COMMAND SUMMARY

OCTAL CODE	MNEMONIC	DESCRIPTION
1003	FNEG	2's complement FACC.
1004	FCLR	Clears FACC.
1005	FINC	Increments FACC.
1006	FSIM	Skips if FACC negative.
1007	FSIP	Skips if FACC positive.
1010	FSIZ	Skips if FACC zero.

7. FLOW CHART

Not Applicable

8. PROGRAM LISTING

Not Applicable